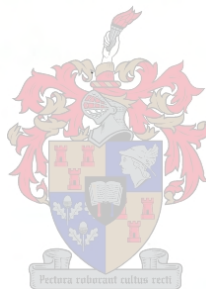


Evaluation of the Constant Current Angle Controlled Reluctance Synchronous Machine drive

Pieter D. Fick



**Thesis presented impartial fulfilment of the requirements
for the degree of Masters of Science in Electrical
Engineering at the University of Stellenbosch**

Supervisor: Prof. M. J. Kamper

March 2002

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

STELLENBOSCH



Abstract

This thesis describes the design and evaluation of a constant current angle controller for a variable speed reluctance synchronous machine (RSM) drive, as an energy efficient high performance drive. An accurate model of the RSM, with the use of finite element analysis, is derived and implemented in simulation software. The current- and speed controllers are designed and evaluated using a complete simulation model of the whole drive. The controller is implemented on a TMS320F240 DSP-based digital controller, which was developed. The dynamic performance of the constant-current-angle control is compared with that of the conventional constant- d -axis-current control method. The results obtained from the RSM drive confirm the simulation results. In the comparison of the two control methods it is shown that the constant-current-angle controlled RSM drive is an energy-efficient drive with good dynamic performance.

Opsomming

Hierdie tesis beskryf die ontwerp en evaluering van 'n konstante stroomhoek beheerder vir 'n reluktansie sinchroon masjien (RSM) aandryfstelsel vir optimum effektiwiteit en dinamika. 'n Akkurate model van die RSM, met behulp van eindige element analise, is opgestel en geïmplimenteer in 'n simulasië pakket. Die stroom- en spoedbeheerders is ontwerp en geëvalueer deur middel van 'n simulasiemodel van die volledige aandryfstelsel. Die beheerder is geïmplimenteer deur gebruik te maak van 'n TMS320F240 DSP-gebaseerde digitale beheerder, wat ontwikkel is. Die konstante-stroomhoek beheer is vergelyk met die konvensionele konstante- d -as-stroom beheer metode. Die resultate van die praktiese stelsel korreleer baie goed met die resultate van die simulasië. Deur die vergelyking van die twee metodes is dit bevind dat die konstante-stroomhoek beheerde RSM aandryfstelsel 'n energie effektiewe aandryfstelsel met baie goeie dinamiese vermoë is.

Acknowledgements

I want to thank the following people who supported me with the completion of this project:

My promoter, Prof. M.J. Kamper, for his assistance, advice, encouragement.

The Electrical Machine Research Group and in particular Jaco Serdyn and Johan Smuts for the assistance and advice with the development of the DSP control system.

The Foundation for Research and Development (FRD) and the University of Stellenbosch for the financial assistance.

The staff of the Electrical Workshop and the Centre for Mechanical Services (SMD) for their assistance with the electrical and mechanical work on the project.

My Parents and family for their encouragement and passions.

Lastly I wish to thank my Lord, without Whom I could not have completed this thesis.

Contents

LIST OF FIGURES.....	IX
LIST OF TABLES.....	XIII
GLOSSARY OF ABBREVIATIONS.....	XIV
LIST OF SYMBOLS.....	XV
1 - INTRODUCTION	1
1.1 BACKGROUND	1
1.2 PROBLEM STATEMENT	2
1.3 APPROACH TO PROBLEM.....	3
1.4 THESIS LAYOUT	4
2 - DYNAMIC MODEL OF THE RSM	6
2.1 REFERENCE FRAMES USED IN POWER SYSTEMS.....	6
2.1.1 <i>Transformation between the abc- reference frame and the dq0- reference frame.....</i>	<i>6</i>
2.1.2 <i>Transformation to the reference frame fixed to the rotor.....</i>	<i>7</i>
2.2 DQO-MODEL OF THE RELUCTANCE SYNCHRONOUS MACHINE	8
2.2.1 <i>dq equivalent circuits.....</i>	<i>8</i>
2.2.2 <i>Torque Equation in dq-variables.....</i>	<i>13</i>
2.2.3 <i>Mechanical model.....</i>	<i>13</i>
2.2.4 <i>Electromagnetic Torque Production</i>	<i>14</i>
3 - POWER CONVERTER AND CONVENTIONAL CURRENT- CONTROL TECHNIQUES	16
3.1 POWER ELECTRONIC CONVERTERS.....	16
3.1.1 <i>Simulation of the full-bridge inverter</i>	<i>17</i>
3.1.2 <i>Transfer function of the inverter.....</i>	<i>18</i>
3.2 CONVENTIONAL CURRENT CONTROL METHODS.....	19
3.2.1 <i>PWM Current Control.....</i>	<i>20</i>
3.2.2 <i>Hysteresis Current Control</i>	<i>21</i>
3.3 SYNCHRONISED CONTROL LOOPS WITH DIGITAL CONTROL	23

4 -	DIGITAL DECOUPLING CURRENT-CONTROL TECHNIQUE FOR THE RSM.....	26
4.1	DESCRIPTION OF THE DECOUPLING CURRENT-CONTROL TECHNIQUE	26
4.2	DESIGN OF THE CURRENT-CONTROLLER	27
4.2.1	<i>Decoupling of the speed voltages</i>	<i>27</i>
4.2.2	<i>Decoupling of the cross-magnetization voltages.....</i>	<i>28</i>
4.2.3	<i>Transfer function of the system.....</i>	<i>29</i>
4.2.4	<i>Root locus design of the PI controllers</i>	<i>31</i>
4.3	IMPLEMENTATION OF THE CURRENT CONTROLLER.....	35
4.3.1	<i>Simulation of the current controller</i>	<i>35</i>
4.3.2	<i>Practical implementation of the current controller.....</i>	<i>38</i>
4.4	SIMULATION AND TESTING OF THE CURRENT CONTROLLER	45
4.4.1	<i>Simulation of the current loops</i>	<i>45</i>
4.4.2	<i>Practical testing of current loops</i>	<i>46</i>
4.4.3	<i>Scaling effect of fixed point processing</i>	<i>48</i>
4.4.4	<i>Decoupling of cross-magnetization</i>	<i>53</i>
4.5	CURRENT CONTROL METHODS	55
4.5.1	<i>Constant d-axis current control method (Conventional Method)</i>	<i>55</i>
4.5.2	<i>Constant current angle control method.....</i>	<i>55</i>
5 -	DIGITAL SPEED-CONTROL OF THE RSM.....	57
5.1	CONSTANT-D-AXIS-CURRENT SPEED CONTROL	57
5.1.1	<i>Constant-d-axis-current controller.....</i>	<i>57</i>
5.1.2	<i>Transfer function of the speed control system</i>	<i>58</i>
5.1.3	<i>Root locus design of the PI speed controller</i>	<i>62</i>
5.2	CONSTANT-CURRENT-ANGLE SPEED CONTROL	63
5.2.1	<i>Constant-current-angle speed controller</i>	<i>63</i>
5.2.2	<i>Transfer function of the speed control system</i>	<i>64</i>
5.2.3	<i>Root Locus Design of the PI speed controller</i>	<i>67</i>
5.3	IMPLEMENTATION OF THE SPEED CONTROLLER.....	68
6 -	HARDWARE OF THE DIGITAL CONTROLLED RSM DRIVE	71
6.1	SUPPLY AND RECTIFIER	71
6.2	INVERTER	72

6.3	ELECTRICAL MACHINE	73
6.4	DIGITAL CONTROLLER.....	74
6.4.1	<i>Processor/Protection Card</i>	75
6.4.2	<i>Measurement Card</i>	78
6.4.3	<i>Position Card</i>	81
6.4.4	<i>Fibre Optic Interface Card</i>	82
7 -	SIMULATION AND PRACTICAL EVALUATION OF THE DYNAMIC CONTROL TECHNIQUES	83
7.1	CONSTANT- <i>D</i> -AXIS-CURRENT SPEED CONTROLLER	83
7.1.1	<i>Testing of the speed controller</i>	84
7.1.2	<i>Load testing of the system</i>	86
7.2	CONSTANT-CURRENT-ANGLE SPEED CONTROLLER	89
7.2.1	<i>Testing of the speed controller</i>	89
7.2.2	<i>Load testing of the system</i>	92
7.3	TORQUE RIPPLE COMPENSATION	94
7.4	COMPARISON OF THE CONTROL METHODS	98
8 -	CONCLUSIONS AND RECOMMENDATIONS	100
8.1	CONCLUSIONS.....	100
8.2	RECOMMENDATIONS	101
	BIBLIOGRAPHY	1
	APPENDIX A – PHOTO ALBUM	108
	APPENDIX B – DSP PROGRAM FOR CONSTANT-<i>D</i>-AXIS-CURRENT SPEED CONTROL	110
	APPENDIX C – DSP PROGRAM FOR CONSTANT-CURRENT-ANGLE SPEED CONTROL	118
	APPENDIX D – EPLD PROGRAM.....	126
	APPENDIX E – FINITE ELEMENT PROGRAM OF RSM.....	129
	APPENDIX F – SCHEMATIC DIAGRAMS OF THE DSP CONTROLLER .	137

List of Figures

FIGURE 2.1 - VECTOR DIAGRAM OF A 2-POLE RSM IN ABC - AND DQO -REFERENCE FRAMES.....	8
FIGURE 2.2 - D -AXIS (A) AND Q -AXIS (B) EQUIVALENT CIRCUITS OF THE RSM.....	9
FIGURE 2.3 - BLOCK DIAGRAM OF DQ -EQUIVALENT CIRCUITS	10
FIGURE 2.4 - D -AXIS FLUX LINKAGE FOR CONSTANT VALUES OF Q -AXIS CURRENT	11
FIGURE 2.5 - Q -AXIS FLUX LINKAGE FOR CONSTANT VALUES OF D -AXIS CURRENT	11
FIGURE 2.6 - (A) D -AXIS SELF-INDUCTANCE (B) Q -AXIS SELF INDUCTANCE VERSUS CURRENT	12
FIGURE 2.7 - (A) D -AXIS MUTUAL INDUCTANCE (B) Q -AXIS MUTUAL INDUCTANCE VERSUS CURRENT.....	12
FIGURE 2.8 - SIMULATION BLOCK DIAGRAM OF RSM-TORQUE AND MECHANICAL SYSTEM.....	14
FIGURE 2.9 - SPACE VECTOR DIAGRAM OF THE RSM.....	14
FIGURE 3.1 - THREE-PHASE HALF-BRIDGE INVERTER TOPOLOGY.....	16
FIGURE 3.2 - THREE-PHASE FULL-BRIDGE INVERTER.....	17
FIGURE 3.3 - SIMULATION BLOCK DIAGRAM OF PHASE-A.....	18
FIGURE 3.4 - TRANSFER FUNCTION OF THE INVERTER	18
FIGURE 3.5 - INVERTER TRANSFER FUNCTION RESULTS	19
FIGURE 3.6 - POWER SYSTEM WITH CURRENT REGULATION	20
FIGURE 3.7 - BLOCK DIAGRAM OF PWM CURRENT REGULATOR	20
FIGURE 3.8 - SIMULATED PWM REGULATOR BY BROD AND NOVOTNY [19].....	21
FIGURE 3.9 - BLOCK DIAGRAM OF THE HYSTERESIS CURRENT REGULATOR	21
FIGURE 3.10 - REFERENCE AND MEASURED CURRENT OF THE HYSTERESIS CURRENT REGULATOR	22
FIGURE 3.11 - SIMULATED HYSTERESIS REGULATOR BY BROD AND NOVOTNY [19]	22
FIGURE 3.12 - BLOCK DIAGRAM OF DSP CONTROL LOOPS	23
FIGURE 3.13 - ACTUAL AND MEASURED CURRENT FOR SYNCHRONISED LOOPS	24
FIGURE 3.14 - ACTUAL AND MEASURED CURRENT FOR UNSYNCHRONISED LOOPS.....	24
FIGURE 3.15 - ILLUSTRATION OF THE SAMPLING OF CURRENT WITH PWM SWITCHING.....	25
FIGURE 4.1 - DECOUPLING CURRENT CONTROL DIAGRAM	26
FIGURE 4.2 - DECOUPLING OF SPEED VOLTAGES.....	28

FIGURE 4.3 - DECOUPLING OF CROSS-MAGNETIZATION AND SPEED VOLTAGES	29
FIGURE 4.4 – BLOCK DIAGRAM OF CURRENT CONTROL SYSTEM IN GENERAL	29
FIGURE 4.5 - (A) STEP RESPONSE AND (B) ROOT-LOCUS OF THE Q -AXIS CURRENT LOOP	32
FIGURE 4.6 - (A) STEP RESPONSE AND (B) ROOT-LOCUS OF THE D -AXIS CURRENT LOOP	34
FIGURE 4.7 - STEP RESPONSE OF THE D -AXIS CURRENT LOOP AT HALF-LOAD.	34
FIGURE 4.8 - SIMULATION BLOCK DIAGRAM OF PI CONTROLLER	35
FIGURE 4.9 - SAMPLING, SCALING AND TRUNCATION OF INPUTS	36
FIGURE 4.10 - SIMUWIN SIMULATION DIAGRAM OF CURRENT CONTROLLER	37
FIGURE 4.11 - PROGRAM DIAGRAM (A) AND FLOW CHART (B) OF THE CURRENT CONTROL PROGRAM.....	38
FIGURE 4.12 – SIMULATED FULL LOAD CURRENT STEP IN THE D -AXIS	45
FIGURE 4.13 – SIMULATED FULL LOAD CURRENT STEP IN THE Q -AXIS	46
FIGURE 4.14 – MEASURED FULL-LOAD CURRENT STEP IN THE D -AXIS WITH $I_Q = 0A$	47
FIGURE 4.15 – MEASURED FULL-LOAD CURRENT STEP IN THE Q -AXIS WITH $I_D = 0A$	47
FIGURE 4.16 - 12-BIT SINE WAVEFORM OF THE LOOKUP-TABLE USED.....	49
FIGURE 4.17 - FIRST TYPE OF SCALING OF VALUES IN INVERSE PARK-TRANSFORMATION	50
FIGURE 4.18 - 10-BIT SINE WAVEFORM OF THE LOOKUP-TABLE USED.....	51
FIGURE 4.19 - IMPROVED SCALING OF THE PARK-TRANSFORMATION.....	52
FIGURE 4.20 – SIMULATED D -AXIS RESPONSE WITH THE FIRST SCALING METHOD	52
FIGURE 4.21 – SIMULATED D -AXIS RESPONSE WITH THE IMPROVED SCALING METHOD	52
FIGURE 4.22 – MEASURED D -AXIS RESPONSE WITH THE FIRST SCALING METHOD.....	53
FIGURE 4.23 – MEASURED D -AXIS RESPONSE WITH THE IMPROVED SCALING METHOD.	53
FIGURE 4.24 - D -AXIS CROSS-MAGNETIZATION VOLTAGE WITH CHANGE IN Q -AXIS CURRENT.	54
FIGURE 4.25 - D -AXIS CURRENT WITH STEP IN Q -AXIS CURRENT WITHOUT DECOUPLING	54
FIGURE 4.26 - D -AXIS CURRENT WITH STEP IN Q -AXIS CURRENT WITH DECOUPLING.....	54
FIGURE 4.27 - SIMULATION BLOCK DIAGRAM OF THE TRANSFORMATION OF THE CURRENT VECTOR TO DQ-CURRENTS.....	56
FIGURE 5.1 – BLOCK DIAGRAM OF SPEED CONTROL LOOP	58

FIGURE 5.2 – MEASURED AND SIMULATED Q -AXIS CURRENT STEP RESPONSE WITH I_D CONSTANT.	59
FIGURE 5.3 – SIMULATED Q -AXIS CURRENT AND DEVELOPED TORQUE STEP RESPONSE WITH CONSTANT D -AXIS CURRENT.	60
FIGURE 5.4 - SPEED LOOP	61
FIGURE 5.5 - (A) STEP RESPONSE AND (B) ROOT-LOCUS OF THE SPEED LOOP	62
FIGURE 5.6 – BLOCK DIAGRAM OF THE SPEED CONTROL LOOP	64
FIGURE 5.7 - STEP IN THE MAGNITUDE OF THE CURRENT VECTOR WITH A CURRENT ANGLE OF 68° (MEASURED AND SIMULATED DQ -CURRENTS)	65
FIGURE 5.8 - STEP IN THE MAGNITUDE OF THE CURRENT VECTOR WITH A CURRENT ANGLE OF 68°	65
FIGURE 5.9 - SIMPLIFIED CONSTANT CURRENT ANGLE SPEED CONTROL	67
FIGURE 5.10 - (A) STEP RESPONSE AND (B) ROOT-LOCUS OF THE SPEED LOOP	68
FIGURE 6.1 - COMPLETE RSM DRIVE	71
FIGURE 6.2 - SUPPLY AND RECTIFIER CIRCUIT	72
FIGURE 6.3 - 3 PHASE FULL-BRIDGE INVERTER	72
FIGURE 6.4 - POWER ELECTRONIC INVERTER AND RECTIFIER	73
FIGURE 6.5 - ROTOR STRUCTURE AND MOTOR DRAWING	73
FIGURE 6.6 - DSP CONTROL UNIT	75
FIGURE 6.7 - DSP CONTROLLER	75
FIGURE 6.8 - LAYOUT OF PROCESSOR/PROTECTION CARD	76
FIGURE 6.9 - LAYOUT OF MEASUREMENT CARD	79
FIGURE 6.10 - CURRENT MEASUREMENT CIRCUIT	79
FIGURE 6.11 - VOLTAGE MEASUREMENT	80
FIGURE 6.12 - LAYOUT OF POSITION CARD	81
FIGURE 7.1 - BASIC BLOCK DIAGRAM OF THE CONSTANT D -AXIS SPEED CONTROL LOOP	83
FIGURE 7.2 - SIMULATED STEP RESPONSE IN SPEED OF THE RSM DRIVE.	84
FIGURE 7.3 - MEASURED STEP RESPONSE IN SPEED OF THE RSM DRIVE.	85
FIGURE 7.4 - SIMULATED RESPONSE OF D - AND Q -AXIS CURRENTS FOR CONSTANT- D -AXIS-CURRENT SPEED CONTROL	85
FIGURE 7.5 - MEASURED RESPONSE OF THE D - AND Q -AXIS CURRENTS FOR CONSTANT- D -AXIS-CURRENT SPEED CONTROL	86

FIGURE 7.6 – CONSTANT-D-AXIS-CURRENT SPEED CONTROL RESPONSE WITH A 0 – 200 NM LOAD STEP.....	87
FIGURE 7.7 - CURRENT REFERENCE (Q -AXIS) DURING STEP IN LOAD TORQUE.....	88
FIGURE 7.8 – CONSTANT-D-AXIS-CURRENT SPEED CONTROL RESPONSE WITH A 200 – 0 NM LOAD TORQUE STEP	88
FIGURE 7.9 - BASIC BLOCK DIAGRAM OF THE CONSTANT-CURRENT-ANGLE SPEED CONTROL	89
FIGURE 7.10 - SIMULATED STEP RESPONSE OF THE SPEED (CONSTANT-ANGLE SPEED CONTROLLER).....	90
FIGURE 7.11 - MEASURED STEP RESPONSE OF THE SPEED (CONSTANT-ANGLE SPEED CONTROLLER).....	91
FIGURE 7.12 - SIMULATED RESPONSE OF D- AND Q-AXIS CURRENTS FOR CONSTANT-CURRENT-ANGLE SPEED CONTROL	91
FIGURE 7.13 - MEASURED RESPONSE OF D- AND Q-AXIS CURRENTS FOR CONSTANT CURRENT ANGLE SPEED CONTROL.....	92
FIGURE 7.14 – CONSTANT-CURRENT-ANGLE SPEED CONTROL WITH A 0 – 200 NM LOAD STEP	92
FIGURE 7.15 – CONSTANT-CURRENT-ANGLE SPEED CONTROL WITH A 200 – 0 NM LOAD STEP.....	93
FIGURE 7.16 - TORQUE RIPPLE OF THE RSM	94
FIGURE 7.17 - BLOCK DIAGRAM OF THE SPEED CONTROLLER WITH TORQUE RIPPLE COMPENSATION	95
FIGURE 7.18 - SIMULATED SPEED WITH (RED) AND WITHOUT (GREEN) RIPPLE COMPENSATION	96
FIGURE 7.19 - Q -AXIS CURRENT WITH (BOTTOM) AND WITHOUT (TOP) RIPPLE COMPENSATION	96
FIGURE 7.20 - SIMULATED SPEED WITH (RED) AND WITHOUT (GREEN) RIPPLE COMPENSATION	97
FIGURE 7.21 – I_s CURRENT WITH (BOTTOM) AND WITHOUT (TOP) RIPPLE COMPENSATION	98

List of Tables

TABLE 5.1 - GAIN OF CONSTANT *D*-AXIS CURRENT CONTROLLER..... 58

TABLE 5.2 - GAIN OF CONSTANT CURRENT ANGLE CURRENT CONTROLLER 66

TABLE 7.1 - CURRENT IN THE RSM..... 99

Glossary of Abbreviations

ac, AC	alternating current
ADC	analog-to-digital converter
DAC	digital-to-analog converter
<i>d</i> -axis	direct axis
dc, DC	direct current
DSP	digital signal processor
EPLD	erasable programmable logic device
FEM	finite element method
IGBT	isolated gate bipolar transistor
IM	induction machine
I/O	input / output
MMF	Magnetomotive force
PI	proportional-integral
PM	permanent magnet
PWM	pulse width modulation
<i>q</i> -axis	quadrature axis
RAM	random access memory
RSM	reluctance synchronous machine
VSI	voltage source inverter
Zoh	zero order hold

List of Symbols

\bar{I}_s, \bar{i}_s	Space vector of stator currents [A \angle rad]
I_s	Magnitude of the stator current vector [A]
\hat{I}_s	Magnitude of the estimated stator current vector [A]
i_{abc}	Instantaneous values of phase currents a, b and c [A]
i_d, i_q	Instantaneous values of d - and q -axis stator currents [A]
I_d, I_q	Steady-state values of d - and q -axis stator currents [A]
\hat{i}_q	Instantaneous value of the estimated q -axis current [A]
J_L	Inertia of the load [kg.m ²]
J_m	Inertia of the machine [kg.m ²]
K_i	Integral constant
K_p	Proportional constant
L_d, L_q	d - and q -axis self-inductance [H]
L'_d, L'_q	Instantaneous values of d - and q -axis self-inductance [H]
M'_d, M'_q	Instantaneous values of d - and q -axis mutual-inductance [H]
P	Number of pole pairs
r_s	Stator resistance
T	Sample period [s]
T_{em}	Electromagnetic torque [Nm]
\hat{T}_{em}	Estimated electromagnetic torque [Nm]
T_L	Load torque [Nm]
\bar{V}_s	Space vector of stator voltage [V \angle rad]
v_{abc}	Instantaneous values of phase voltages a, b and c [V]
v_d, v_q	Instantaneous values of d - and q -axis stator voltages [V]
β_L	Friction factor for the load
β_m	Friction factor for the machine
$\bar{\lambda}_s$	Space vector of stator flux linkage [Wb \angle rad]
λ_s	Magnitude of the flux linkage vector in the stator [Wb]

λ_{abc}	Instantaneous values of phase flux-linkages a, b and c [Wb]
λ_d, λ_q	d - and q -axis flux linkages of the stator [Wb]
$\bar{\lambda}_l$	Space vector of stator fundamental airgap flux linkage [Wb \angle rad]
$\bar{\lambda}_m$	Space vector of stator leakage flux linkage [Wb \angle rad]
γ	Angle between the current vector and the flux linkage vector [rad]
τ_s	2% Settling time [s]
ϕ	Current angle (angle between d -axis and current vector) [rad]
φ, θ	Rotor angle (angle between the magnetic a -axis and the q -axis [rad]
θ_e, θ_{re}	Electrical rotor angle [rad]
θ_m, θ_{mr}	Mechanical rotor angle [rad]
ω	Angular velocity of the arbitrary reference frame [rad/s ⁻¹]
ω_{re}, ω_e	Electrical angular velocity of the rotor [rad/s ⁻¹]
ω_{mr}, ω_{rm}	Mechanical angular velocity of the rotor [rad/s ⁻¹]
ξ	Damping factor

1 - Introduction

There is always the quest for better dynamic performance with the least amount of energy used when it comes to moving things from one place to another. This is definitely the case in the field of transport. From the beginning of the last century the motor vehicle manufacturers has constantly improved the design and control of the petrol and diesel fuelled machines to get the most power and torque and have the best fuel economy.

In the last half of the previous century electrical machine drives were started being used in transport. These include trains and in the last few decades military vehicles and, with the higher fuel price amongst other things, motor vehicles.

Together with traction applications there is also another field of application that needs high dynamic performance with optimum efficiency. This is in the field of servo drives which needs fast responses to move or re-orientate objects quickly and accurately.

1.1 Background

Nowadays there are a lot of different types of electrical machines. They can be divided into two main groups, namely the AC machines and the DC machines. The most commonly used machines are the DC machine and the induction AC machine. Extensive research and development have been done on the dynamic control of these two machines, using various power electronics and digital controllers in variable speed drives.

There are also a few not so familiar or unconventional electrical machines. The Reluctance Synchronous Machine (RSM) is one of these machines and is a single-salient reluctance machine. The RSM consists of a standard, non-salient 3-phase stator and an unexcited, salient rotor. With the development of the semi-conductor

technology and variable speed drives, this machines' performance became more competitive with respect to that of the induction machine. There are studies done to compare the performance of these two machines by Vagati [25] and Fratta [26] and also by Germishuizen [1] and Kamper [13] of the University of Stellenbosch.

The qualities needed for the electrical machine drive are:

- high torque density
- small torque ripple
- low total losses, thus high efficiency
- good dynamic torque response

Taking these qualities into account the RSM drive is a good choice because of its good torque density, high efficiency and good dynamic performance [5].

1.2 **Problem statement**

To effectively control the RSM, the correct positioning of the stator current vector, with respect to the rotor, is necessary. With vector control the RSM is able to develop much higher torque. This can only be done by position and current feedback to a controller in a closed-loop drive system.

Much research has been done on ways to control the current. One of the most commonly used methods is constant d -axis current control. More advanced methods, like given in [2]-[10], were developed to improve the control for better dynamics and efficiency in the RSM drive system. A few of them is summarised briefly as follows:

- In [3] an optimum-efficiency control strategy is proposed by Matsuo, El-Antably and Lipo by using variable d -axis current control. This means the d -axis current is used to ensure flux in the machine and the magnitude of the q -axis current is controlled to control the developed torque. Together with this the controller also consists of an optimum efficiency controller. This controller monitors the input power of the RSM drive and adjusts the d -axis current, and thus also the flux in the machine, to find the optimum efficiency operating

point for generating a specific torque. This control is also a complex control method and it has more input variables than the normal control methods.

- Vagati in [4] researched high dynamic performance control, by evaluating the losses in the machine, the relation between the d - and q -axis current and the error in measurement of the current vector. After considering these areas, a constant d -axis control scheme is proposed which makes use of a flux observer for the control of the d -axis flux linkage. The flux observer is less sensitive to disturbances, like saturation, than that of a normal d -axis current controller. The q -axis current controller controls the developed torque. This method, due to the fact that there is constant field current and constant flux present in the machine, is not energy efficient, especially at standstill or under no-load conditions.

These proposed methods consist of different ways of optimising the current control, minimise losses and input power for the same or better output torque. The dynamics of the controller is also improved. The problem is how does the constant current angle control method compare in dynamic performance and efficiency with the conventional constant d -axis current control. Therefore the constant current angle control method is evaluated, implemented and compared with constant d -axis current control.

1.3 Approach to problem

To investigate constant current angle control an accurate simulation and practical implementation is necessary. From finite element analysis the flux and inductance parameters of the RSM in dq -quantities are determined. For this the focus is on a 42kW RSM, optimum designed by Kamper [13]. The effects of cross-magnetization and saturation on the parameters of the RSM are taken into account and used in the simulation of the machine. The RSM is fully modelled in the simulation.

Classical control system design is done of the dq digital current controllers and also the speed controller. The effects of cross-magnetization and speed voltages on these

current controllers are minimized by matter of decoupling. The power electronic converter with PWM switching and digital sampling is implemented in simulation and a complete and accurate simulation of the whole drive is developed.

The whole control system is implemented on a DSP digital controller that is developed and from accurate simulation and practical measurement the performances of the different controllers are evaluated.

1.4 Thesis layout

The layout for the remainder of this thesis is as follows:

- Chapter 2: The dynamic dq -model of the Reluctance Synchronous Machine is derived and discussed in depth.
- Chapter 3: The type and modelation of the power converter is described. The conventional current-control methods and their advantages / disadvantages discussed.
- Chapter 4: The digital decoupling-current-control technique for the RSM drive is proposed in this chapter. The difference between the floating point and fixed-point simulation and implementation of the digital control is also discussed.
- Chapter 5: The digital constant-current-angle speed control and the constant-d-axis-current speed control of the RSM are described and evaluated.
- Chapter 6: This chapter explains the hardware that is developed and used to implement the current and speed control strategies on a RSM drive.

Chapter 7: The practical and simulated results of the current- and speed control are shown and discussed. The torque ripple compensation method is simulated on two speed controllers and results are discussed. The widely used constant- d -axis-current and constant-current-angle control methods are evaluated and compared.

Chapter 8: In this chapter a summary with conclusions are given and recommendations are made for further research in this area.

2 - Dynamic model of the RSM

The Reluctance Synchronous Machine has an unexcited salient rotor, which is symmetrical and does not consist of any conductors. Thus the only circuits to consider in the analysis of the RSM are the stator winding circuits. The three phase circuits each have a resistance, self-inductance and also a mutual inductance with the other two phase- circuits. The self- and mutual inductances vary with the position of the rotor. There are thus three differential voltage equations that define the three phase circuits. In this chapter the electrical and mechanical model of the RSM in the dq0-reference frame is explained.

2.1 Reference frames used in power systems

The most general reference frame that is used when describing three-phase systems is the three-phase variable- or *abc*- reference frame [11]. The reason for this is that this frame describes the system variables in real time. Besides this reference frame there is another frame, called the arbitrary- or *dqo*- reference frame [11]. This reference frame is more compatible for complex control systems on electrical machines than the *abc*-reference frame.

2.1.1 Transformation between the *abc*- reference frame and the *dqo*-reference frame

A change in variables from the *abc*- reference frame to the *dqo*- reference frame is accomplished through the following equation:

$$\overline{f}_{qd0s} = \overline{K}_s \overline{f}_{abcs} \quad (2.1)$$

where the variables in this equation are described as:

$$\left(\overline{f}_{qd0s} \right)^T = \begin{bmatrix} f_{qs} & f_{ds} & f_{os} \end{bmatrix} \quad (2.2)$$

$$\left(\overline{f}_{abcs} \right)^T = \begin{bmatrix} f_{as} & f_{bs} & f_{cs} \end{bmatrix} \quad (2.3)$$

where T implies the transpose of the vector.

The transformation matrix is given by:

$$\overline{K}_s = \frac{2}{3} \begin{bmatrix} \cos \varphi & \cos(\varphi - \frac{2\pi}{3}) & \cos(\varphi + \frac{2\pi}{3}) \\ \sin \varphi & \sin(\varphi - \frac{2\pi}{3}) & \sin(\varphi + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad (2.4)$$

where

$$\varphi = \int_0^t \omega_{re}(\xi) d\xi + \varphi(0) \quad (2.5)$$

and ξ is a temporary variable of the integration.

The inverse transformation is done through the following equation:

$$\overline{f}_{abcs} = (\overline{K}_s)^{-1} \overline{f}_{qdos} \quad (2.6)$$

where:

$$(\overline{K}_s)^{-1} = \begin{bmatrix} \cos \varphi & \sin \varphi & 1 \\ \cos(\varphi - \frac{2\pi}{3}) & \sin(\varphi - \frac{2\pi}{3}) & 1 \\ \cos(\varphi + \frac{2\pi}{3}) & \sin(\varphi + \frac{2\pi}{3}) & 1 \end{bmatrix} \quad (2.7)$$

The variables f can represent current, voltage or flux linkage. The subscript s implies that the variables, parameters and equations are associated with stationary circuits.

2.1.2 Transformation to the reference frame fixed to the rotor

This reference frame rotates synchronously with the rotor and eliminates all the time varying inductances in the machine equations [11]. This makes it easier for the design of the control system, because the signals of the control loops are dc quantities in the steady state. This transformation is known as the Park's transformation. For this transformation the following is defined:

$$\varphi = \int_0^t \omega_{re}(\xi) d\xi = \theta, \quad (2.8)$$

where ω_{re} is the electrical angular velocity of the rotor, φ the rotor angle or angle between the q -axis on the rotor and the magnetic a -axis of the stator. The angle φ is equal to θ as shown in Fig. 2.1.

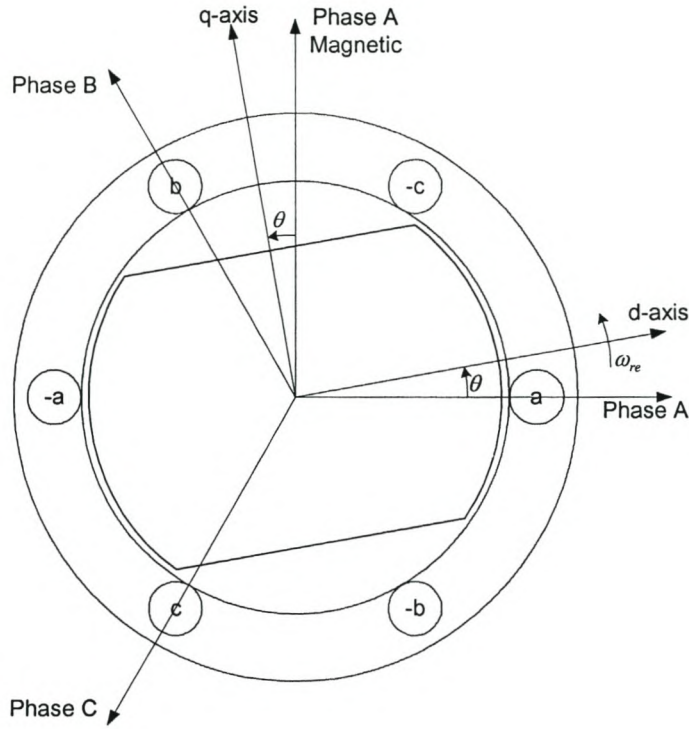


Figure 2.1 - Vector diagram of a 2-pole RSM in *abc*- and *dqo*-reference frames

2.2 *dqo*-model of the Reluctance Synchronous Machine

In this section the machine model of the RSM is defined in the *abc*-reference frame and transformed to the *dqo*-reference frame. The *dq*-equivalent circuit as well as the torque and mechanical circuit is derived and discussed.

2.2.1 *dq* equivalent circuits

The Reluctance Synchronous Machine has a normal 3-phase stator. The general equation for the stator voltage in the *abc*-reference frame is given as:

$$\bar{v}_s = r_s \bar{i}_s + \frac{d\bar{\lambda}_s}{dt}, \quad (2.9)$$

with

$$\bar{v}_s = \begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix}; \quad \bar{i}_s = \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}; \quad \bar{\lambda}_s = \begin{bmatrix} \lambda_a \\ \lambda_b \\ \lambda_c \end{bmatrix}$$

This equation can be transformed to the dqo -reference frame using equation (2.1) and with the result the following set of equations are formed:

$$v_d = r_s i_d + \frac{d\lambda_d}{dt} - \omega_{re} \lambda_q \quad (2.10)$$

$$v_q = r_s i_q + \frac{d\lambda_q}{dt} + \omega_{re} \lambda_d \quad (2.11)$$

$$v_0 = r_s i_0 + \frac{d\lambda_0}{dt} \quad (2.12)$$

where r_s is the phase-resistance and ω_{re} the electrical velocity. If a balanced 3-phase system is used, v_0 is zero, thus equation (2.12) can be leaved out. The other two equations can be visualized through the circuits of Fig. 2.2, which are the dq -equivalent circuits of the RSM.

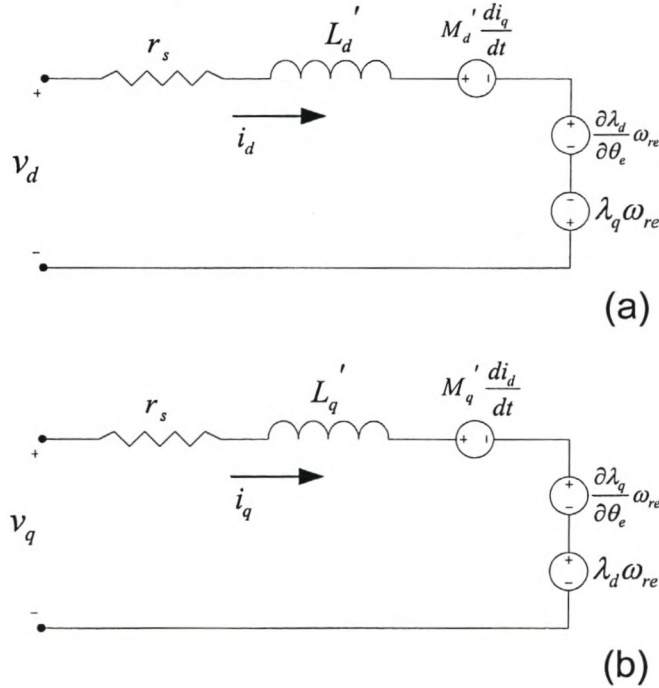


Figure 2.2 - d -axis (a) and q -axis (b) equivalent circuits of the RSM

The d -axis and q -axis flux linkages are both functions of the d -axis current, q -axis current and the rotor angle, thus

$$\lambda_d = f(i_d, i_q, \theta) \quad \text{and} \quad \lambda_q = f(i_d, i_q, \theta).$$

This implies that:

$$\begin{aligned}\frac{d\lambda_d}{dt} &= \frac{\partial \lambda_d}{\partial i_d} \frac{di_d}{dt} + \frac{\partial \lambda_d}{\partial i_q} \frac{di_q}{dt} + \frac{\partial \lambda_d}{\partial \theta} \frac{d\theta}{dt} \\ &= L_d' \frac{di_d}{dt} + M_d' \frac{di_q}{dt} + \frac{\partial \lambda_d}{\partial \theta} \omega_{re}\end{aligned}\quad (2.13)$$

and

$$\begin{aligned}\frac{d\lambda_q}{dt} &= \frac{\partial \lambda_q}{\partial i_q} \frac{di_q}{dt} + \frac{\partial \lambda_q}{\partial i_d} \frac{di_d}{dt} + \frac{\partial \lambda_q}{\partial \theta} \frac{d\theta}{dt} \\ &= L_q' \frac{di_q}{dt} + M_q' \frac{di_d}{dt} + \frac{\partial \lambda_q}{\partial \theta} \omega_{re}\end{aligned}\quad (2.14)$$

Skewing the rotor laminations by one stator slot pitch can reduce the influence of the positioning of the rotor on the flux linkage. Although the machine used does not have a skewed rotor, the model is simplified by not incorporating this component into the model. Thus from equations (2.13) and (2.14) together with equations (2.10) and (2.11) the dq -equations of the stator voltages of the RSM becomes:

$$v_d = r_s i_d - \omega_{re} \lambda_q + L_d' \frac{di_d}{dt} + M_d' \frac{di_q}{dt} \quad (2.15)$$

$$v_q = r_s i_q + \omega_{re} \lambda_d + L_q' \frac{di_q}{dt} + M_q' \frac{di_d}{dt} \quad (2.16)$$

The two qd -equivalent circuits in Fig. 2.2 visualize equations (2.15) and (2.16). These voltage equations are implemented in a block diagram in the simulation program as is shown in Fig. 2.3.

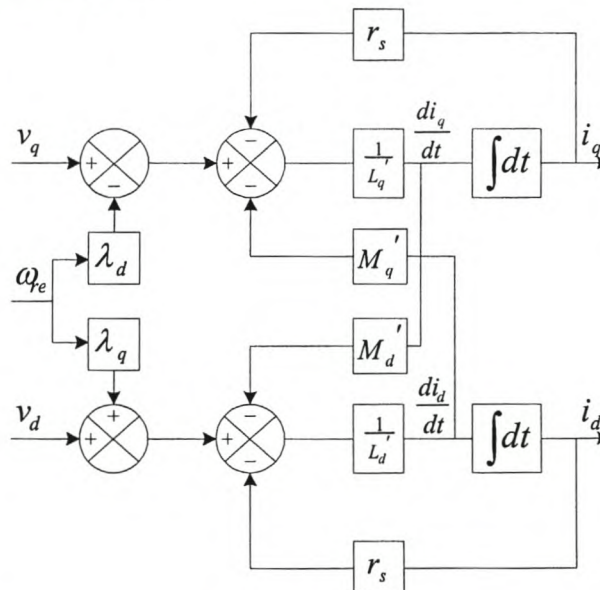


Figure 2.3 - Block diagram of dq -equivalent circuits

The stator resistance, r_s , is constant, but the d - and q -axis inductances, L_d' , L_q' , M_d' and M_q' , is only constant in the steady state and not in the dynamic model of the RSM. As shown earlier the d - and q -axis flux linkages is dependent on the d - and q -axis current. Using two-dimensional Finite Element Analysis program [13], given in appendix E, the d - and q -axis flux linkages is calculated and is given in Fig. 2.4 and 2.5 respectively.

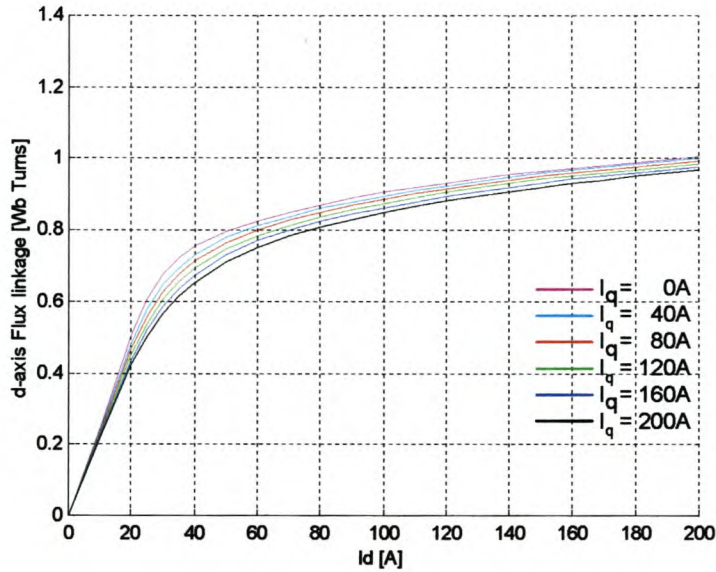


Figure 2.4 - d -axis flux linkage for constant values of q -axis current

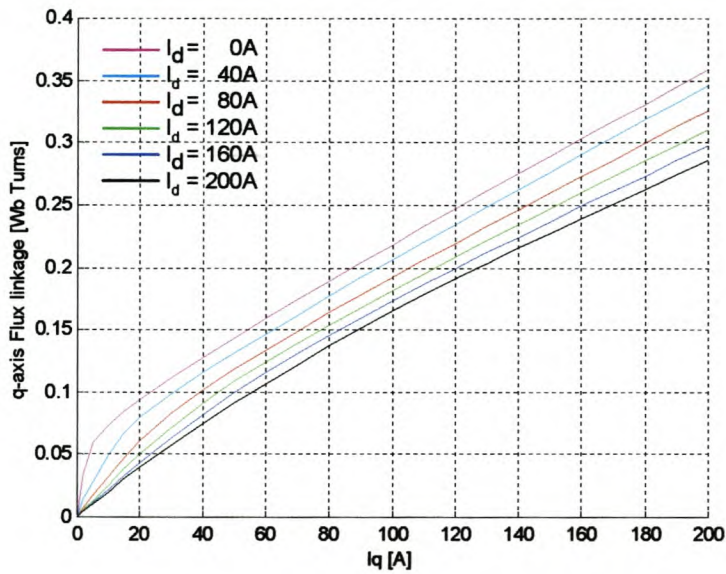


Figure 2.5 - q -axis flux linkage for constant values of d -axis current

From equations (2.13) and (2.14) the d - and q -axis self-inductances are defined as

$$L_d' = \frac{\partial \lambda_d}{\partial i_d} \text{ and } L_q' = \frac{\partial \lambda_q}{\partial i_q}. \quad (2.17)$$

These equations are used together with the results in Figs. 2.4 and 2.5 to calculate the d - and q -axis self-inductances and are given in Fig. 2.6 below.

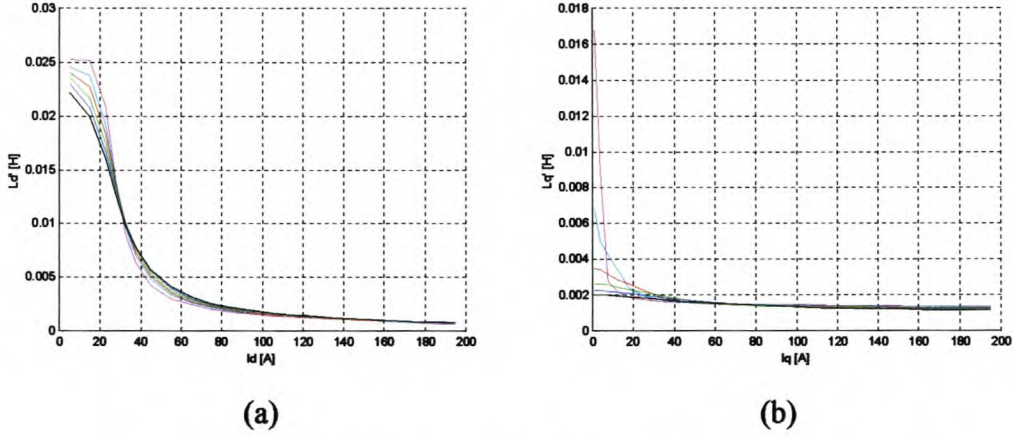


Figure 2.6 – (a) d -axis self-inductance (b) q -axis self inductance versus current

Equations (2.13) and (2.14) also defines the d - and q -axis mutual inductances [15] as

$$M_d' = \frac{\partial \lambda_d}{\partial i_q} \text{ and } M_q' = \frac{\partial \lambda_q}{\partial i_d}. \quad (2.18)$$

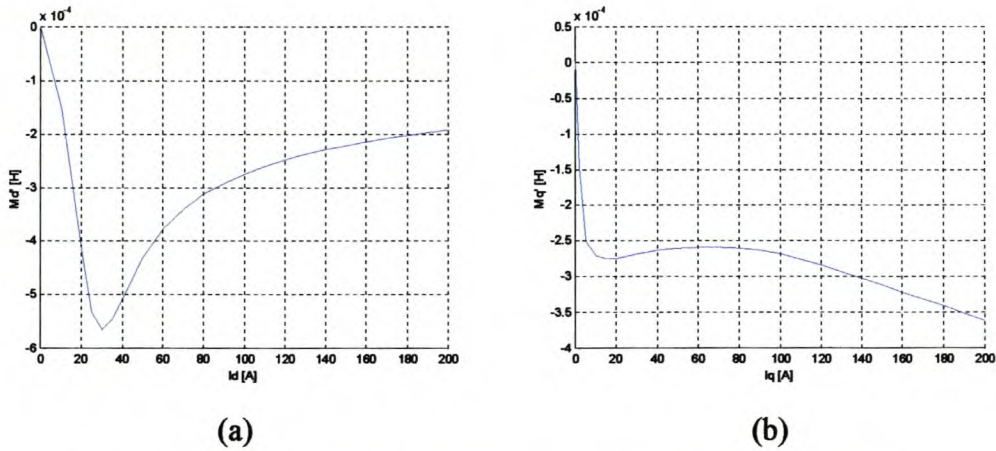


Figure 2.7 – (a) d -axis mutual inductance (b) q -axis mutual inductance versus current

These results is now converted to lookup tables and used in the blocks for λ_q , λ_d , M_q ,

M_d , $\frac{1}{L_q}$ and $\frac{1}{L_d}$ in the diagram. These lookup tables have got two inputs that are not

shown on Fig 2.3. The d - and q -axis currents are the reference for these lookup tables.

2.2.2 Torque Equation in dq -variables

In the previous section the two dq -equations for the voltage were derived. To complete the dq -model of the RSM the equation for the torque as a function of the current and the flux linkage must be defined. This equation can be derived from the power. The complete equation for the torque is given by [12] as:

$$T_{em} = \frac{3}{2} \frac{p}{2} \left[(\lambda_d i_q - \lambda_q i_d) + \frac{1}{2} \left(i_d \frac{\partial \lambda_d}{\partial \theta} + i_q \frac{\partial \lambda_q}{\partial \theta} \right) \right] \quad (2.19)$$

This equation is further simplified by neglecting the variation of the flux linkage versus rotor position as discussed earlier. Therefore

$$T_{em} = \frac{3}{2} \frac{p}{2} (\lambda_d i_q - \lambda_q i_d). \quad (2.20)$$

2.2.3 Mechanical model

The equation for the mechanical system is given by [26] as

$$T_{em} = J_{eq} \frac{d\omega_{rm}}{dt} + \beta_{eq} \omega_{rm} + T_L \quad (2.21)$$

where T_{em} is the torque of the RSM according the equation (2.20), T_L is the load torque, J_{eq} and β_{eq} is the inertia and the friction respectively and ω_{rm} the mechanical speed of the machine.

The inertia and friction can be written as follows:

$$J_{eq} = J_m + J_L \quad (2.22)$$

$$\beta_{eq} = \beta_m + \beta_L \quad (2.23)$$

where J_m and J_L are the inertia of the machine and the load respectively. Also β_m and β_L are the friction of the machine and the load respectively.

The relation between the mechanical speed and the electrical speed is:

$$\omega_{rm} = \frac{1}{p} \omega_{re} \quad (2.24)$$

and the electrical angle of the rotor can be determined by integrating the electrical angular velocity, thus

$$\theta_e = \int \omega_{re} dt \quad (2.25)$$

The equations for the torque, speed and rotor position are combined in the simulation block diagram of Fig. 2.7.

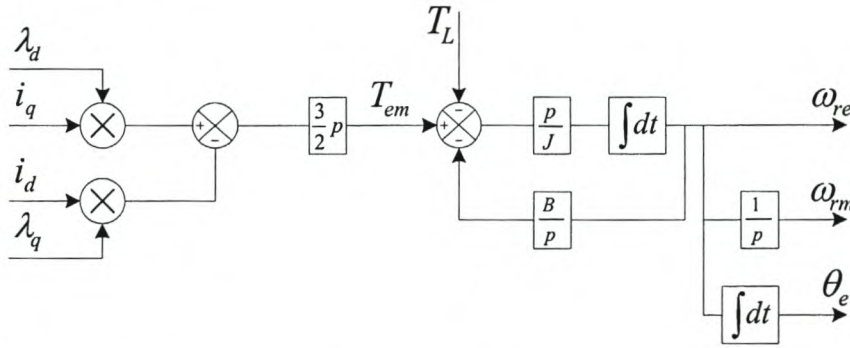


Figure 2.8 – Simulation block diagram of RSM-torque and mechanical system.

2.2.4 Electromagnetic Torque Production

Consider the space vector diagram of the RSM shown in Fig. 2.8. In the diagram the current and flux linkage space vectors are shown. The current space vector can be placed at any position with respect to the d -axis of the rotor. Because of the difference in reluctance of the rotor at different angles the ideal position for the rotor is to be aligned with the MMF wave. The reason for this is that the minimum reluctance flux path is through the d -axis. If the stator MMF is placed in such a way that the rotor is not aligned, the rotor is going to move to align and torque is developed.

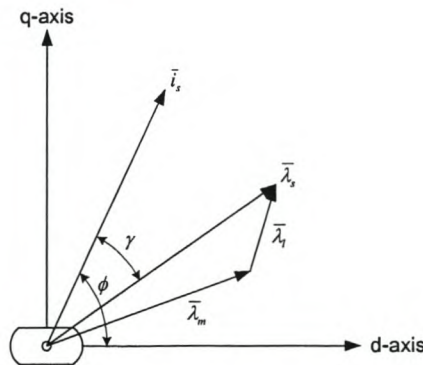


Figure 2.9 – Space vector diagram of the RSM

The magnitude of the torque produced is proportional to the cross product of the fundamental flux linkage vector and the current vector.

$$T = \|k\bar{\lambda}_s \times \bar{i}_s\| \quad (2.26)$$

where k is a constant. Eqn. (2.24) can be rewritten as

$$T = k\lambda_s I_s \sin(\gamma) \quad (2.27)$$

with λ_s and I_s are the magnitudes of the stator flux linkage and the current space vector respectively. Equations. (2.24 and 2.25) can be expressed in terms of the dq -components as

$$T = \frac{3}{2} p (\lambda_d I_q - \lambda_q I_d) \quad (2.28)$$

where λ_d and λ_q are the d- and q-axis fundamental stator flux linkage components and I_d and I_q the fundamental stator current components. The d- and q-axis inductances L_d and L_q can be defined as

$$L_d = \lambda_d / I_d \text{ and } L_q = \lambda_q / I_q \quad (2.29)$$

and eqn (2.26) can be expressed as

$$T = \frac{3}{2} p (L_d - L_q) I_d I_q \quad (2.30)$$

or

$$T_{em} = \frac{3}{4} p (L_d - L_q) I_s^2 \sin(2\phi) \quad (2.31)$$

where ϕ is the angle of the stator current vector.

If L_d and L_q are constant in eqn (2.31), the maximum torque is produced at a current angle ϕ of 45° . This is according to theory however, because of non-linearity like saturation and cross-magnetization, the current angle for the maximum torque is higher. Through experimental testing and simulation the angle is typical between 60° - 70° .

3 - Power Converter and Conventional Current-Control Techniques

From the mathematical model of the RSM, as given in Chapter 2, it is clear that the RSM can be controlled by controlling the frequency and amplitude of the 3-phase supply voltage. This can be done in the open loop mode, without feedback of the rotor position or current, by slowly varying the supply frequency. This can only be used for no-load or small load conditions.

However to have good dynamic performance the current vector must be controlled with respect to the position of the rotor. To effectively control the current vector the voltage vector have to be controlled and therefore a power electronic converter is used. Together with the converter a current controller is needed. There are also various methods of current control. This chapter will consider a few conventional current control techniques as well as certain obstacles encountered with these techniques. Firstly the power electronic converter topologies that can be used are considered.

3.1 Power Electronic Converters

There are basically two types of dc-to-ac voltage source converters. The one commonly used is the 3-phase half-bridge inverter. This inverter consists of three phase-arms and a dc-link capacitor bank as shown in Fig 3.1.

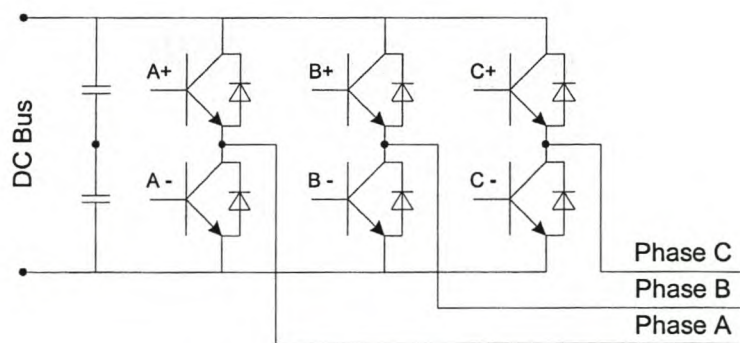


Figure 3.1 - Three-phase half-bridge inverter topology

This topology has the advantage that only 3 phase-terminals of the machine and three cables are necessary. The machine can be connected in delta or in star and in the star connection the star point is not used. One of the limits is that with this topology only half of the dc-bus voltage on average can be applied over a phase during switching.

Another topology that can be used is the full-bridge inverter. This inverter uses six phase-arms instead of the three for the half-bridge inverter. The inverter is shown in Fig. 3.2.

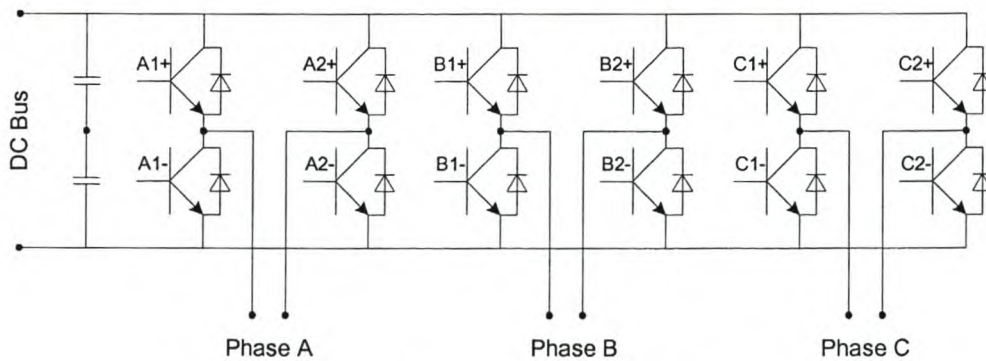


Figure 3.2 - Three-phase full-bridge Inverter

There are three important differences of this inverter compared to the half-bridge inverter. The first is that each phase is controlled independently. The second difference is that the full bus-voltage can be applied to a phase winding.

An important advantage is that unipolar switching can be used instead of bipolar switching. With unipolar switching the virtual switching frequency is double that of the actual switching frequency. This gives a lower current ripple than with the lower switching frequency. This converter was available in the laboratory and is used in the project for the dynamic control of the RSM.

3.1.1 Simulation of the full-bridge inverter

The full-bridge inverter configuration of Fig. 3.2 with unipolar switching is implemented in the project. With unipolar switching the two phase-arms of the full-bridge is operating independent of each other. The voltage over a phase can be positive V_{dc} , negative V_{dc} or zero. This enables the machine to apply a switching frequency over the phase winding of double the switching frequency applied to the power electronic switches.

Unipolar switching is applied by comparing the reference voltage to a triangle wave, with a frequency that of the switching frequency, and if the reference value is higher than the triangle wave at any instance the output is high, otherwise the output is low. This result is the PWM signal for the first phase-arm. The PWM signal for the other phase-arm of the same phase is generated in the same way but the negative of the reference voltage is compared with the triangle wave. This switching strategy for one phase is described by the block diagram as shown in Fig. 3.3. This block diagram is used in the simulation of the inverter. The dc-bus voltage in this project is about 550V and the switching frequency is 3.33kHz.

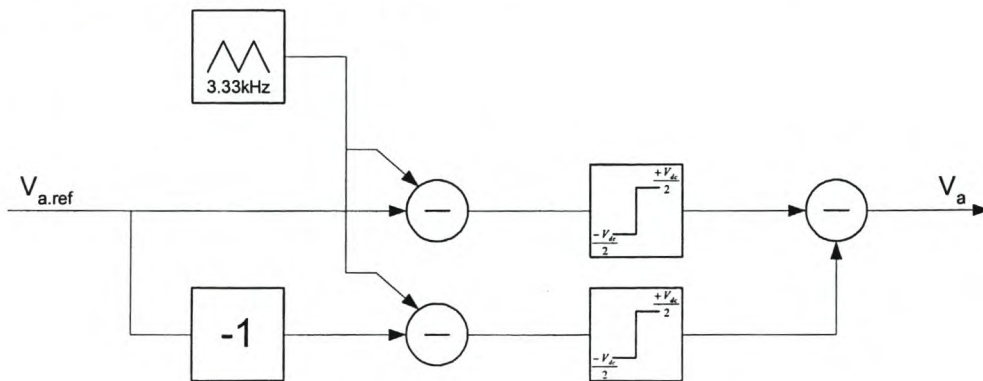


Figure 3.3 – Simulation block diagram of phase-a

3.1.2 Transfer function of the inverter

If the effects of the time delay and switching frequency of the converter are ignored, then the transfer function of the converter can be simplified by means of a constant, as shown in Fig. 3.4. Both the delay and the switching frequency have practically no effect in general on the control system.

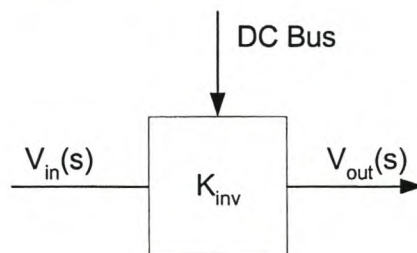


Figure 3.4 - Transfer function of the inverter

To obtain this transfer function and in particular the value of K_{inv} , the output voltage V_{out} is simulated as well as measured on the actual drive system for different values of the input voltage reference V_{in} . The simulated and measured output voltage versus the input voltage reference in the DSP controller is given in Fig. 3.5. The scale of the input voltage reference is as it is implemented in the DSP.

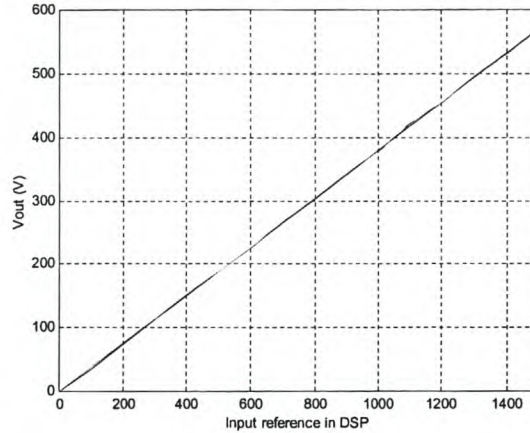


Figure 3.5 - Inverter transfer function results

From this results it can be seen that the simulated and the measured results is close to each other. The simulated model is in fact a good representation of the actual drive. The small difference between the simulated and measured results are because of the assumption made about the time delay and switching frequency.

From the results in Fig 3.5 the values of K_{inv} is given as obtained in each case as:

$$K_{inv.sim} = 0.381 \text{ and } K_{inv.prac} = 0.377$$

Thus K_{inv} is taken as 0.377 in the design of the controllers of the RSM system as described in the next chapter.

3.2 Conventional Current control methods

There are a few basic or conventional methods for controlling the current in electrical machine drive systems. Fig. 3.6 shows a block diagram of a three-phase power converter system with current regulation. The two most commonly used methods are the PWM and hysteresis current regulators. These methods are briefly described in the following sub-sections. There are also thousands of other more advanced methods.

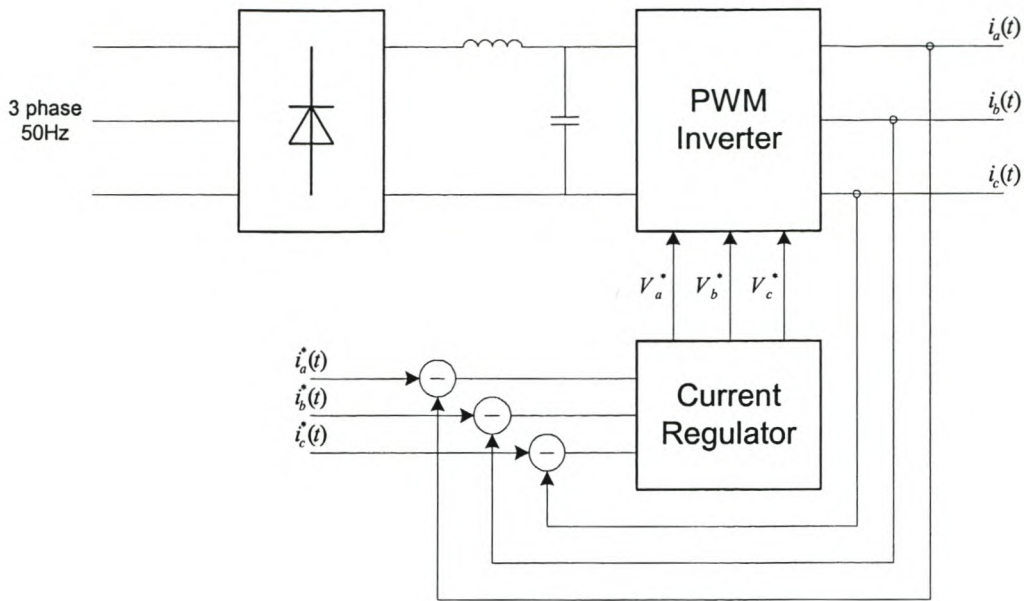


Figure 3.6 – Power system with current regulation

3.2.1 PWM Current Control

The first current control technique discussed in this section is the PWM current regulator that uses a constant switching frequency. A block diagram of the PWM regulator is shown in Fig. 3.7.

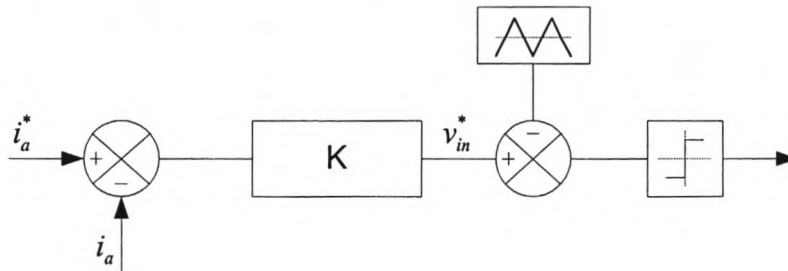


Figure 3.7 – Block diagram of PWM current regulator

As is shown in Fig. 3.7 the measured current, i_a , is subtracted from the reference current i_a^* to obtain the error in the current. This error is multiplied by the constant K and the voltage v_{in}^* is compared with a fixed frequency and amplitude triangular wave. If the magnitude of the control voltage is less than that of the triangular wave the phase arm is switched to positive V_{DC} in the case of a full-bridge inverter. Otherwise the phase arm is switched to the negative of V_{DC} .

This controller is applied to each of the phases of the system. A PI controller can also replace the constant K in Fig. 3.7. A simulation of the PWM current regulator done by Brod and Novotny [18] is shown in Fig. 3.8. The regulator that is implemented is a K controller. This result shown that the measured current has a phase as well as a magnitude difference with respect to the reference current. The magnitude and phase difference can be minimized by increasing the value of K (see Fig. 3.7). Care must be taken with the choice of K , because the control can become unstable for large values of K . By using a PI controller in place the constant K the magnitude and phase difference can be improved, but there will still be a error because of the ac waveform of i_a^* .

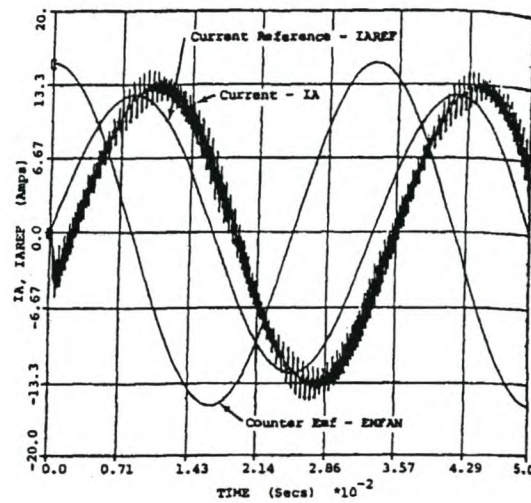


Figure 3.8 - Simulated PWM regulator by Brod and Novotny [19]

3.2.2 Hysteresis Current Control

The hysteresis current regulator is also a well-known and very simple current regulator. A block diagram of the hysteresis current regulator is given in Fig. 3.9.

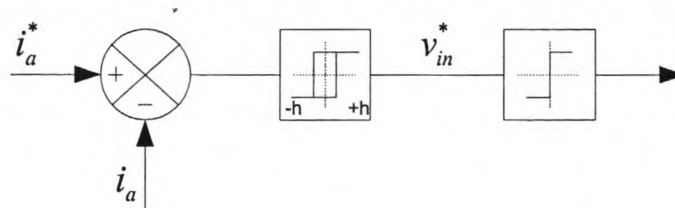


Figure 3.9 - Block diagram of the Hysteresis current regulator

The error between the reference current and the measured current is put through a hysteresis switch. Looking at Fig. 3.10 it can be seen that a hysteresis band of magnitude h is around the reference current. If the measured current reaches either of these limits, the output of the hysteresis switch changes. This method is also called a bang-bang current regulator.

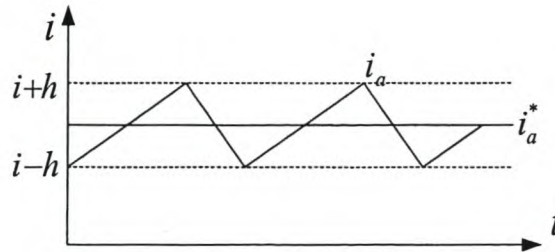


Figure 3.10 - Reference and measured current of the hysteresis current regulator

The regulator is simulated amongst other by Brod and Novotny in [19] and the result is given in Fig. 3.11. The switching frequency of this current regulator is not constant as in the case of the PWM current regulator, as can be seen in Fig. 3.11. This creates a problem, because in a digital control system it is necessary to have a constant switching frequency for measuring the ac currents. In some instances the switching frequency of the measured current is much higher than the frequency of the control loop. This is a good regulator if implemented as an analog controller, but not that well suited for a digital controller.

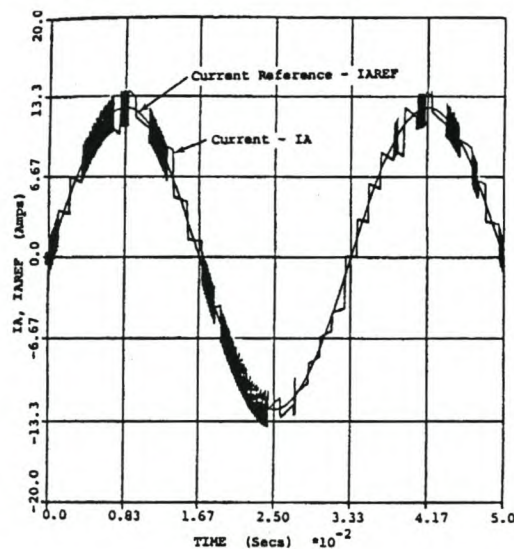


Figure 3.11 - Simulated hysteresis regulator by Brod and Novotny [19]

There are thus disadvantages in both methods with respect to the error between the measured and reference currents. In Chapter 4 a current control method is proposed

that have good dynamic response with almost no error in current. The method also decouples the cross-magnetization between the phases.

3.3 Synchronised Control Loops with Digital Control

In digital current control it is important to measure, in a switching cycle, the average or fundamental value of the current. This can be done by

- (i) synchronising the program control loop (and the sampling of the currents) with the triangular wave of the PWM generator, and
- (ii) sampling the current at the right instant in a switching cycle.

A third requirement is to sample the current at the instant when no power switching is taking place, to avoid noise measurement.

In the DSP controller there are basically two control loops running. The one is the control program loop, which consists of all the measurements and control decisions. The second control loop running is the PWM generator, which receives the voltage references from the control program loop as shown in Fig. 3.12.

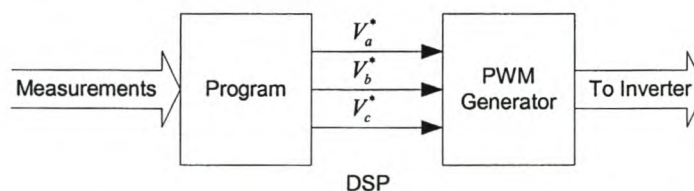


Figure 3.12 - Block diagram of DSP control loops

Looking at the synchronising of the program control loop and the triangular wave of the PWM generator, the actual and sampled current of a RL circuit is simulated. The results in Fig. 3.13, of the two loops that are synchronised, show the actual current (blue) being sampled on its fundamental component by the measured current (green).

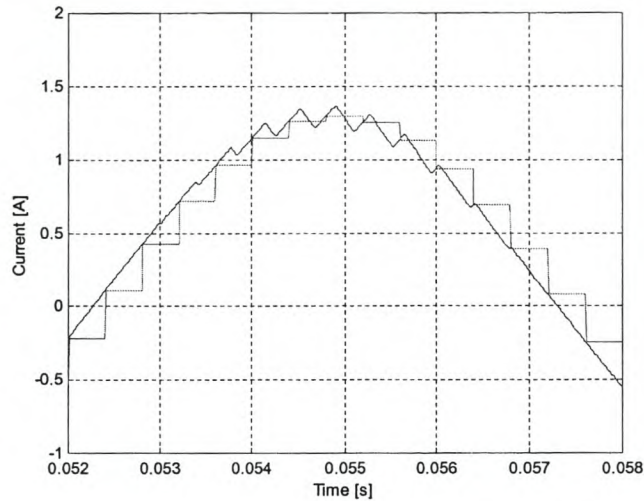


Figure 3.13 - Actual and measured current for synchronised loops

Looking now at Fig. 3.14, which is the results of the two loops that are not synchronized, it can be seen that especially at the peak of the actual current wave (blue) the measured current differs a lot from the fundamental of the actual current.

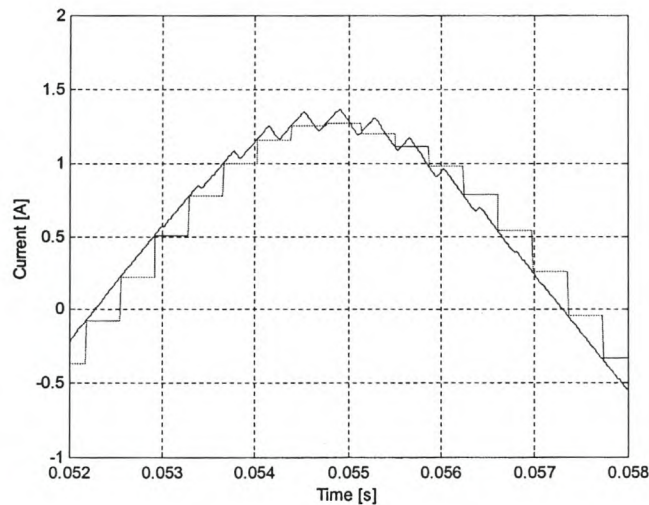


Figure 3.14 - Actual and measured current for unsynchronised loops

Now, considering the right instant to measure the current, the best time is when the triangle wave of the PWM generator is at a maximum or a minimum, because switching occurs between these two time points and the sampling is on the fundamental components of the current.

In Fig. 3.15 the triangle wave reference of the PWM generator (black), the reference and actual currents (red) and the measured current (blue) are shown. In this figure sampling is done at the minimum value of the triangle waveform. If the switching frequency is much higher than the base frequency of the ac current, then for a small part of the current waveform the waveform can be considered as constant, as assumed in Fig. 3.15. The minimum point of the triangle waveform will always be in the middle of the switching points and the measured value will be on the fundamental of the actual current.

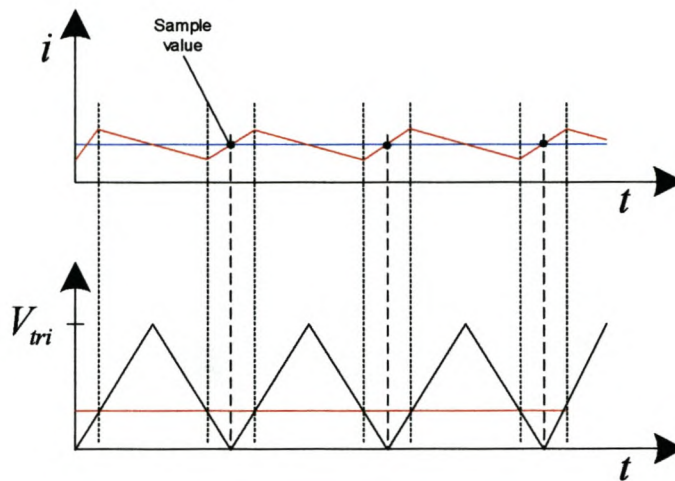


Figure 3.15 - Illustration of the sampling of current with PWM switching.

4 - Digital decoupling current-control technique for the RSM

In the previous chapter the conventional current-control techniques are discussed. In this chapter a decoupling current-control technique is used for the RSM drive. This technique is applied in the $dq0$ - reference frame and consists of amongst other things proportional-integral controllers and decoupling of the speed-voltages. The layout, design and implementation of this control technique in the digital controller are covered.

4.1 Description of the decoupling current-control technique

The current control technique is implemented as mentioned in the $dq0$ -reference frame. The three phase currents of the machine are transformed, using the Park-transform. These d - and q -axis currents are then subtracted from the d - and q - axis reference currents (see Fig. 4.1). The PI controllers react on the errors between the reference and actual currents to control the currents in the dq -phases. Adding or subtracting the cross-magnetization and speed-voltages from the output voltage signals of the PI controllers decouple the control from the interferences of cross-magnetization and speed voltage fluctuation. The output voltages are then transformed back to the abc - reference frame and applied to the PWM inverter to power the machine. The block diagram of this control strategy is given in Fig. 4.1.

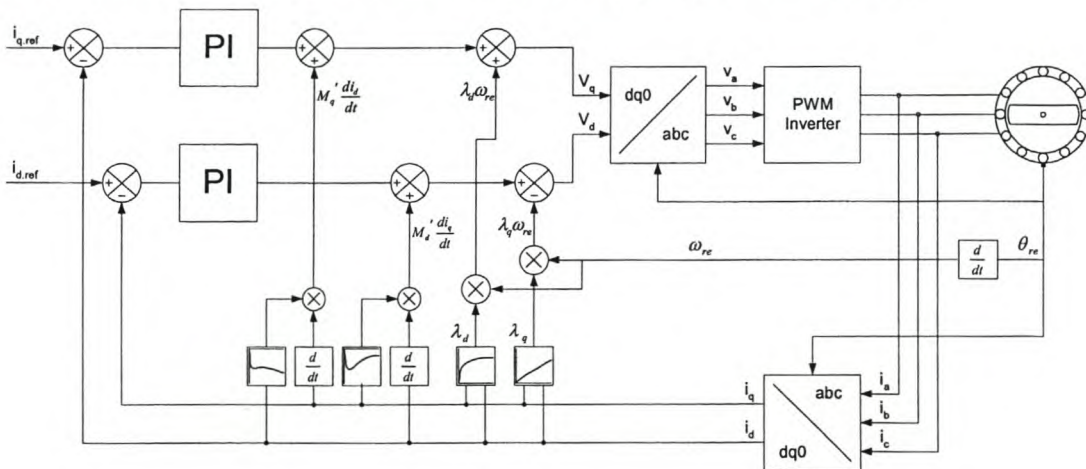


Figure 4.1 - Decoupling current control diagram

4.2 Design of the current-controller

There are a few aspects that have to be considered in the design of the current controllers. The decoupling of the various voltages mentioned in the previous section and the design of the two PI controllers are discussed in this section. The last part of this section explains the implementation of the current loops the simulation software as well as on the actual system itself.

4.2.1 Decoupling of the speed voltages

There are a few variables that can have an influence on the operation of the current controllers. Referring back to equation (2.15) and (2.16) there are two disturbances that affects the performance of the current controllers. The first of these disturbances are the speed voltages. A speed voltage is given by the product of the electrical speed and the flux linkage of the opposite axis. This means that the speed voltages are dependent on the speed of the machine and during acceleration these voltages affect the total voltage necessary to control the current. Equation (2.15) and (2.16) can be rewritten as:

$$(v_d + \omega_{re}\lambda_q) = \left(r_s i_d + L'_d \frac{di_d}{dt} \right) + M'_d \frac{di_q}{dt} \quad (4.1)$$

$$(v_q - \omega_{re}\lambda_d) = \left(r_s i_q + L'_q \frac{di_q}{dt} \right) + M'_q \frac{di_d}{dt} \quad (4.2)$$

In the case of the control of the q -axis current, the q -axis voltage is affected by $\omega_{re}\lambda_d$, while in the case of the d -axis current, the d -axis voltage increasing with $\omega_{re}\lambda_q$. If the speed voltages can be decoupled from the current control loop, the PI controllers will only respond to the currents through the stator resistance, self-inductance and mutual inductance. Adding or subtracting these speed voltages from the output voltages of the PI controllers will decouple the control from the speed voltages.

The d -axis current loop is given in Fig. 4.2(a). After the PI controller the speed voltage is subtracted and then inside the machine model the speed voltage is added again. This ideally cancels the speed voltage in the current control loop. The same is done for the q -axis as shown in Fig. 4.2(b).

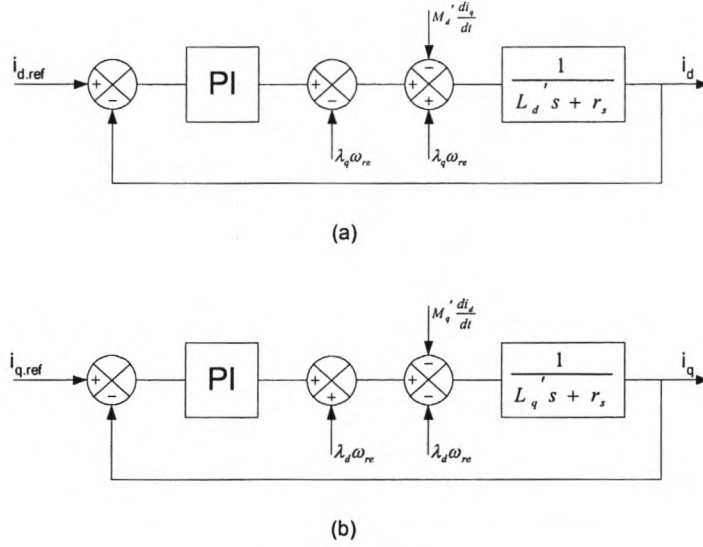


Figure 4.2 - Decoupling of speed voltages

4.2.2 Decoupling of the cross-magnetization voltages

From equations 4.1 and 4.2 it can be seen that with the speed voltages removed, there are still terms left that will cause disturbances in the current control. These are the cross-magnetization terms. The two equations can be rewritten as:

$$\left(v_d + \omega_{re} \lambda_q - M'_d \frac{di_q}{dt} \right) = \left(r_s i_d + L'_d \frac{di_d}{dt} \right) \quad (4.3)$$

$$\left(v_q - \omega_{re} \lambda_d - M'_q \frac{di_d}{dt} \right) = \left(r_s i_q + L'_q \frac{di_q}{dt} \right). \quad (4.4)$$

According to these equations the cross-magnetization terms must be added in the controller to compensate for the subtracting of them in the machine model. This is shown in the diagram of Fig. 4.3. As with the speed voltages, the cross-magnetization terms can be ideally cancelled out, or else the PI-current controllers are decoupled from the effect of these terms.

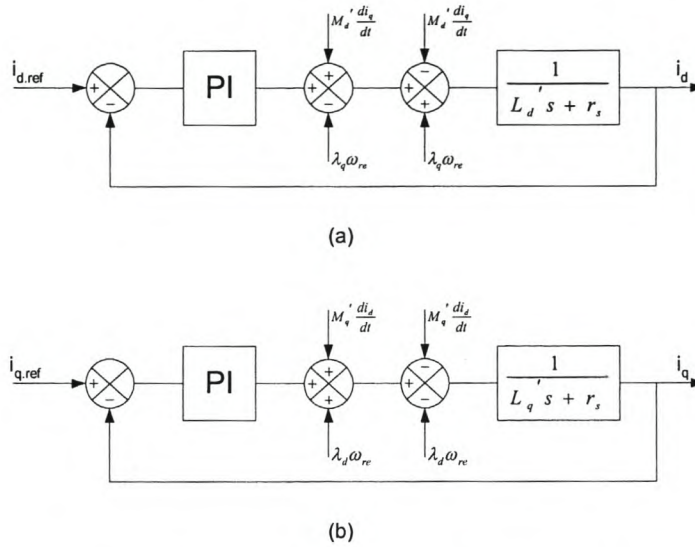


Figure 4.3 - Decoupling of cross-magnetization and speed voltages

4.2.3 Transfer function of the system

As seen from the previous section the speed voltages and the mutual inductances can be ignored in the design of the PI controllers because they are decoupled and have no influence on the transient performance of the current control system. The controller is digital and therefore the PI controllers are designed in the discrete-time system or z-plane. With this in mind the block diagram for the each of the current loops is shown as in Fig. 4.4.

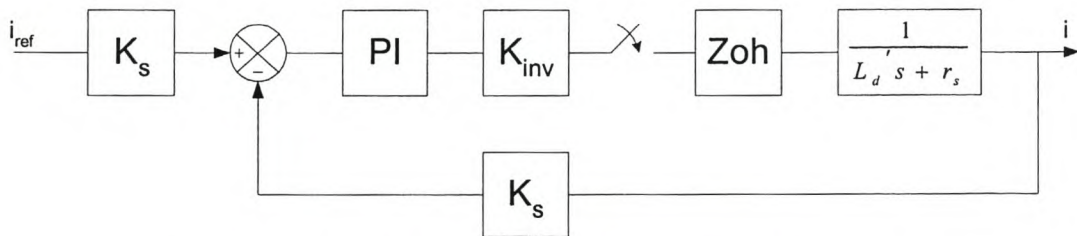


Figure 4.4 – Block diagram of current control system in general

Because of the digital design a sampler and Zoh block are added to the loop. K_{inv} is the constant that represent the transfer function of the inverter and K_s is the scaling constant for the current signals into the fixed-point digital controller.

In the continuous-time system or s-plane the transfer function of the PI controller is given as:

$$D(s) = K_p + \frac{K_i}{s} \quad (4.5)$$

This function can be transformed to the discrete-time system by using the backward difference method. The relationship between the s-plane and the z-plane using this method is $s = \frac{z-1}{zT}$ where T is the sample period. Using this relationship, the PI controller in the discrete-time system can be written as follows:

$$D(z) = \frac{(K_p + TK_i)z - K_p}{z - 1} \quad (4.6)$$

The transfer function of the system and the Zoh circuit must be jointed in the s-plane and then transformed to the z-plane. The transfer function of the Zoh block is given by:

$$G_{Zoh}(s) = \frac{1}{s}(1 - e^{-sT}) \quad (4.7)$$

and that of the system by:

$$G_s(s) = \frac{1}{L's + r_s} \quad (4.8)$$

where L' is the phase self-inductance. The transfer function of the system and Zoh circuit is thus from equations 4.7 and 4.8 the following:

$$G(s) = (1 - e^{-sT}) \left(\frac{1}{s} \right) \left(\frac{1}{sL' + r_s} \right). \quad (4.9)$$

This transfer function is transformed to the discrete-time system and is given as follows:

$$G(z) = \frac{1 - e^{-\frac{r_s T}{L'}}}{r_s \left(z - e^{-\frac{r_s T}{L'}} \right)} \quad (4.10)$$

The total closed loop transfer function of the current loop is calculated by using equations 4.6 and 4.10 as follows:

$$G_{cl}(z) = \frac{\frac{K_{inv}K_s}{r_s} \left[1 - e^{-\frac{r_s T}{L'}} \right] \left[(K_p + TK_i)z - K_p \right]}{z^2 + \left[\left(-1 - e^{-\frac{r_s T}{L'}} \right) + \frac{K_{inv}K_s}{r_s} (K_p + TK_i) \left(1 - e^{-\frac{r_s T}{L'}} \right) \right] z + \left[e^{-\frac{r_s T}{L'}} - \frac{K_p K_{inv}K_s}{r_s} \left(1 - e^{-\frac{r_s T}{L'}} \right) \right]} \quad (4.11)$$

This equation is used for the calculation of the yet unknown variables of the two PI controllers.

4.2.4 Root locus design of the PI controllers

The diagram of the current loop as given by Fig. 4.4 can be used for both the d -axis and q -axis PI controllers. These two controllers are designed separately in the following sub-sections.

a. Q -axis PI current loop

The open loop transfer function of the current loop is derived from equation (4.6) and (4.10). The parameters used is as follows:

$$\begin{aligned} R_s &= 48.9\text{m}\Omega \text{ (stator resistance of the RSM)} \\ T &= 300\mu\text{Sek. (sampling period)} \\ K_s &= 1.7067 \text{ (conversion factor for DSP)} \\ K_{inv} &= 0.377 \text{ (transfer function of inverter)} \end{aligned}$$

The outstanding parameter that is needed is a steady state value of the q -axis self-inductance. This value is chosen at the full-load operation point of the q -axis self-inductance versus current curve shown in Fig. 2.5, because the inductance is almost constant over the whole region except at very low currents. The rated current of the machine is 110A rms. Thus with $I_q = 141\text{A}$ and $I_d = 66\text{A}$ (Current angle = 68°), the q -axis inductance is:

$$L_q' = 1.37\text{mH (steady state } q\text{-axis self-inductance at full-load)}$$

Now the open loop transfer function can be calculated as:

$$G_{ol}(z) = D(z)(K_s K_{inv})G(z)$$

$$= D(z) \frac{0.1401}{z - 0.9894} \quad (4.12)$$

Using the transfer function, a root locus design is done to determine the transfer function of the PI controller. This design is done with the help of MATLAB for the following specifications:

$$\tau_s \leq 2 \text{ milliseconds (settling time)}$$

$$\xi > 1 \text{ (over damped damping factor)}$$

The root locus is shown in Fig. 4.5(b) together with the unit step response of the system in Fig. 4.5(a).

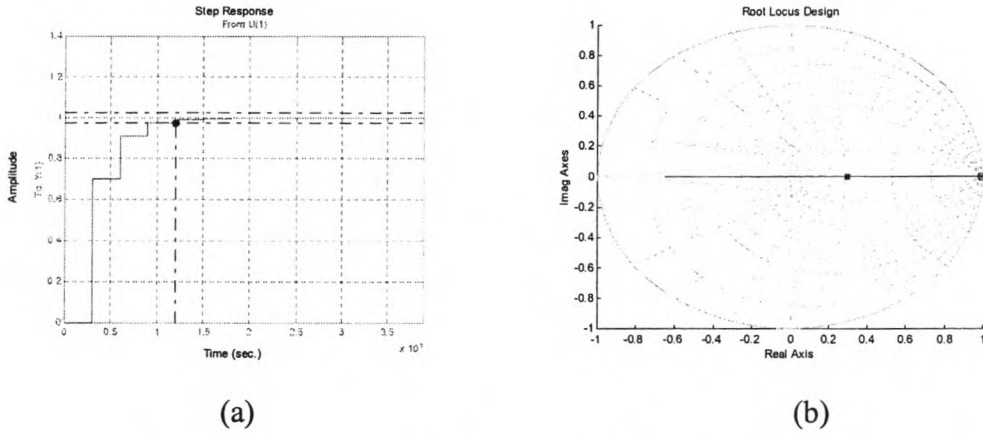


Figure 4.5 - (a) Step Response and (b) Root-locus of the q -axis current loop

The system has a settling time of 1.21 milliseconds and is over damped (see Fig. 4.5(a)). From the root locus the gain of the PI controller as well as the position of the pole and zero is obtained (see Fig. 4.5(b)). These values are given in the transfer function of the PI controller in equation (4.13) below:

$$D(z) = \frac{5(z - 0.99)}{z - 1}$$

$$= \frac{5z - 4.95}{z - 1} \quad (4.13)$$

and the values of proportional and integral constants are calculated as:

$$K_p = 4.95 \text{ and } K_i = 166.67$$

for the q -axis PI current controller.

b. *D*-axis PI current loop

Most of the parameters used for the design of the *d*-axis PI controller are the same as for the *q*-axis PI design, with the exception of the *d*-axis self-inductance. The self-inductance that is used is chosen again at full-load. The curve for the *d*-axis self-inductance in Fig. 2.5 is not linear, but around the full-load the inductance is taken as almost constant. With this in mind the response in the full-load region will be designed fast so that the response in the other working regions are satisfactory. The self-inductance in the full-load region is (see Fig. 2.5):

$$L_{d,\min}' = 2.5\text{mH (Steady-state } d\text{-axis self-inductance at full-load)}$$

The self-inductance at halve-load is also looked at to evaluate the response in the lower load region. This inductance is (see Fig. 2.5):

$$L_{d,\max}' = 12\text{mH (Steady-state } d\text{-axis self-inductance at halve load)}$$

The open loop transfer function is calculated again as:

$$\begin{aligned} G_{ol}(z) &= D(z)(K_s K_{inv})G(z) \\ &= D(z) \frac{0.07695}{z - 0.9942} \end{aligned} \quad (4.14)$$

With this transfer function a root-locus design is used to calculate the transfer function constants of the PI controller. The design is done in MATLAB, as is done for the *q*-axis control system. The specifications are as follows:

$$\tau_s \leq 1\text{milliseconds (settling time)}$$

$$\xi > 1 \text{ (over damped)}$$

The root-locus plot and the step response of the system are shown in Fig. 4.6.

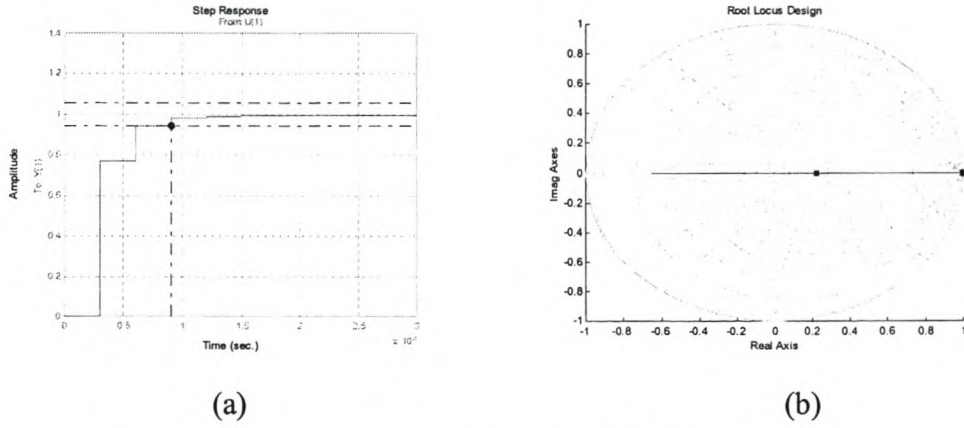


Figure 4.6 - (a) Step response and (b) Root-locus of the d -axis current loop

The system has a settling time of about $900\mu\text{s}$ which is just less than the 1ms mark. It is also over damped. This design is done with the d -axis self-inductance taken at full-load. The controller's response is also evaluated at halve-load to see what the effect of the variation in inductance is. The step response for this is shown in Fig. 4.7.

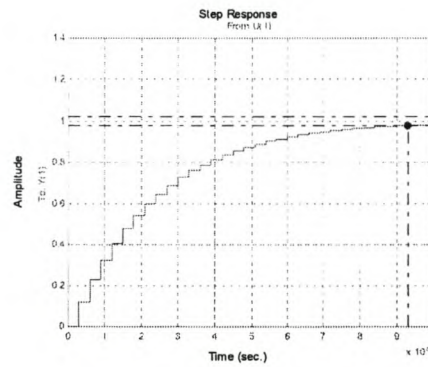


Figure 4.7 - Step response of the d -axis current loop at half-load.

This response will affect the step response of the actual system at low current because of the non-linearity in the d -axis inductance. From the root locus the gain of the PI controller as well as the position of the pole and zero are obtained (see Fig. 4.6(b)). These values are given in the transfer function of the PI controller in equation 4.15 below:

$$D(z) = \frac{10(z - 0.999)}{z - 1}$$

$$= \frac{10z - 9.99}{z - 1} \quad (4.15)$$

and the values of proportional and integral constants are calculated as:

$$K_p = 9.99 \text{ and } K_i = 33.333$$

for the d -axis PI current controller.

4.3 Implementation of the current controller

The current controller is implemented in two ways. First by simulation of the controller and the whole drive system and secondly by implementing in an actual drive system using a fixed-point DSP digital controller. For the simulation the SIMUWIN software, developed by the University of Stellenbosch, is used. In the simulation the controller is implemented in exactly the same way as in the actual controller so that actual results can be compared with simulated results.

4.3.1 Simulation of the current controller

There are a few sets of components in the simulation of the dq current-controller. The first set of components is the transfer functions of the dq PI controllers. These transfer functions, as derived in the design section of this chapter, are realised in the z -plane. The basic block diagram for these transfer functions is shown in Fig. 4.8.

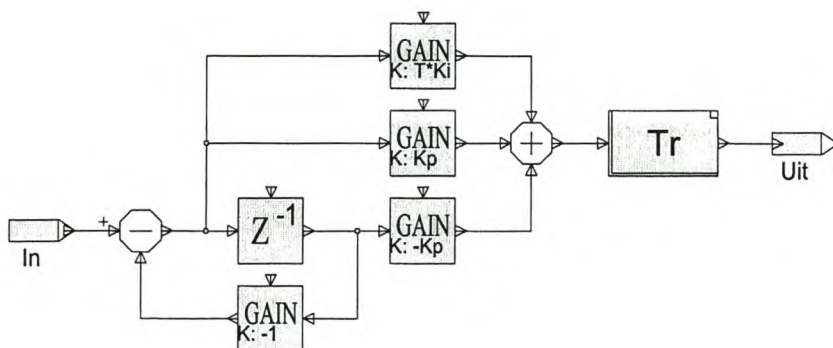


Figure 4.8 - Simulation block diagram of PI controller

The block diagram is realised from equation (4.6) where K_p is the proportional constant, K_i the integral constant and Z^{-1} the unit delay at the sampling rate.

The second set of components is the scaling of the feedback currents, reference currents, speed and angle from their actual magnitudes to magnitudes that can be handled within the digital controller. This also includes the sampling of these inputs and due to the use of a fixed point DSP controller for the drive system because of the truncation of the inputs and other signals.

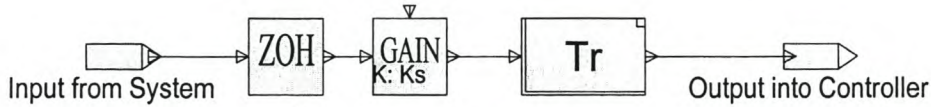


Figure 4.9 - Sampling, scaling and truncation of inputs

A Zoh block is used for the sampling and is set at the sampling frequency, that is also equal to the switching frequency. Following the Zoh block is the scaling factor of the DSP for the current. This factor is calculated as follows:

$$K_s = \frac{\text{Maximum of } A/D \text{ output}}{\text{Maximum real current}}$$

$$= \frac{512}{300A} = 1.7067 \quad (4.16)$$

The last block truncates the value, in other words, it converts the value from a floating-point value to a fixed-point value.

The third set of components is the park- and the inverse park-transform block as discussed in Chapter two. These two blocks is used to transform the currents and voltages into the appropriate reference frame.

The total diagram for the controller is as follows:

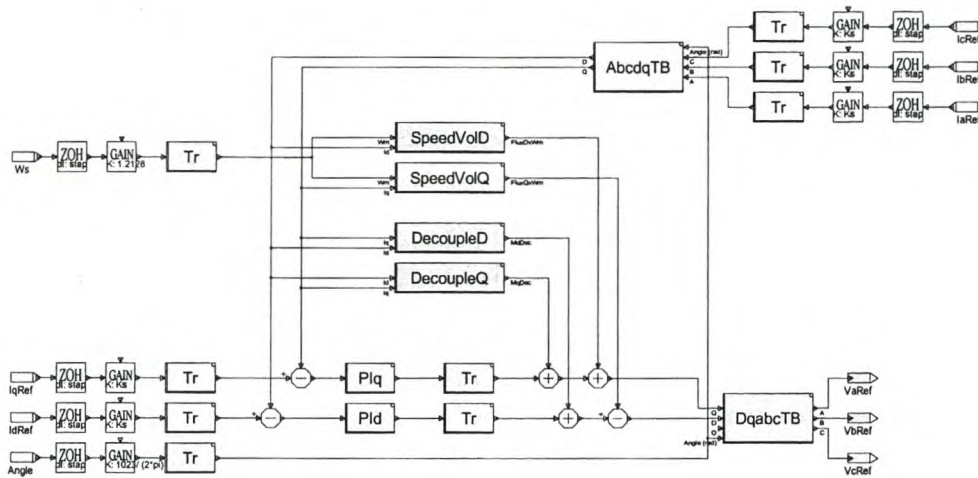


Figure 4.10 - Simuwin simulation diagram of current controller

The inputs on the left of Fig. 4.10 are the reference currents, electrical angle and the speed of the machine. These inputs are scaled as discussed earlier. The inputs in the top-right corner are the actual measured currents of the machine and these are also scaled. The currents are in the *abc*-reference frame and are therefore transformed to the *dq0*-reference frame by means of the Park-transformation implemented in the *AbcdqTB*-block. These measured *dq*-currents are then subtracted from the *dq*-reference currents and the error values are given to the two PI controllers implemented in the *PI_q*- and *PI_d*-blocks as discussed earlier in the section.

In the middle of the diagram the speed-voltages and cross-magnetization terms are calculated and decoupled from the system at the outputs of the PI controllers. The speed-voltages are calculated using two look-up tables that were generated from the finite-element results given in Chapter two. The curves used to generate the look-up tables for each of the *dq*-currents are chosen at the rated current of the opposite *dq*-current.

The cross-magnetization blocks also consist of two look-up tables determining M_d' and M_q' , that is then multiplied by the derivative of the *dq*-currents as expressed in equations (4.1) and (4.2) and shown in Fig. 4.3.

4.3.2 Practical implementation of the current controller

The current-controller is also implemented in an actual RSM drive system using a fixed-point DSP controller. The hardware of the controller is described in detail in Chapter 6. The controller measures the 3-phase currents of the machine and converts these to digital values to be used by the processor. The position and speed are measured digitally using a resolver and then read by the processor. The rest of the control is implemented in software running in the DSP processor.

The program is written in ANSI C-code. As with the simulation implementation discussed in the previous section, the separate parts of the controller are described and also how these are programmed into the source program. The diagram of the controller program and the flow chart are shown in Fig. 4.11.

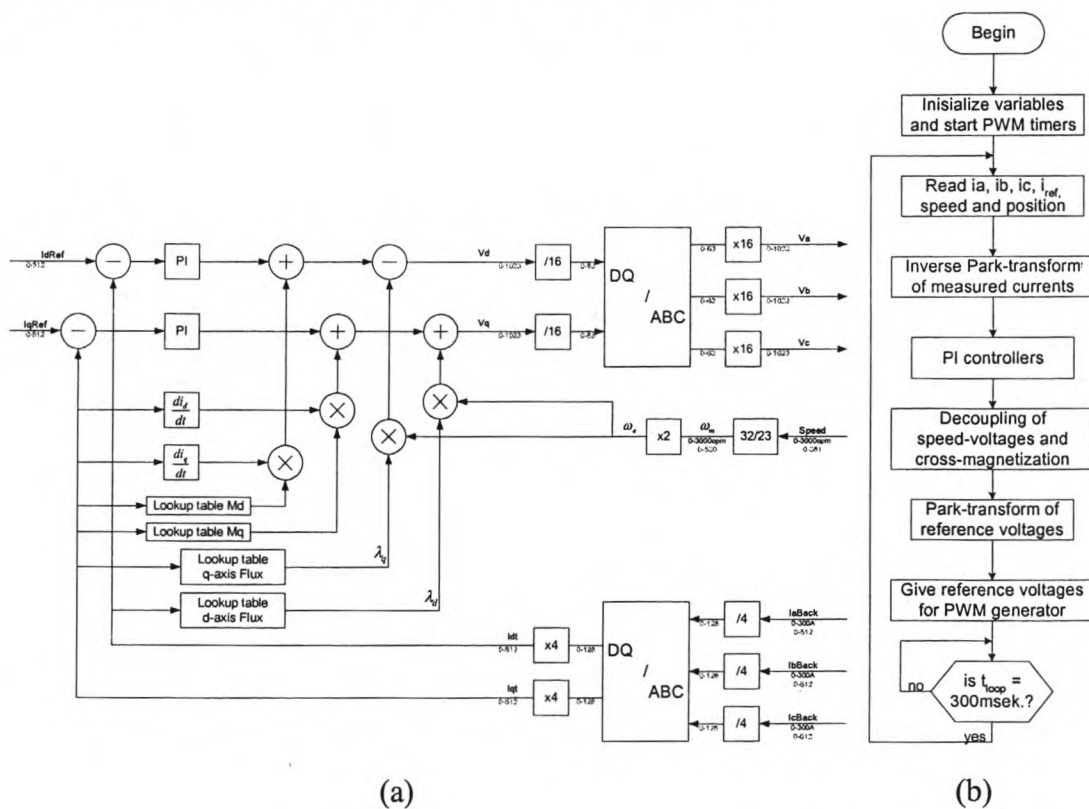


Figure 4.11 - Program diagram (a) and flow chart (b) of the current control program

The main part of the program is an infinite for-loop in which all the tasks are done in one sampling period. This loop-time is the sampling time of the controller and starts on the lowest value of the triangle wave used for the PWM generation. The sampling rate is equal to the switching frequency of the PWM generator.

The first task of the control program is the reading of the measured values starting with the three phase currents, the speed and then the current reference command. This part includes the scaling of the measured values, which is necessary because of the method of measurement and the type of analog-to-digital conversion that is used. The currents are measured with a hall effect “LEM” module together with an analog circuit, before being converted to a digital value within the DSP. The constant for this conversion is:

$$\begin{aligned}
 \text{Value inside DSP} &= \left(\frac{\text{Current reference (V)}}{\text{Actual Current}} \right) \left(\frac{\text{Digital Value}}{\text{Current reference (V)}} \right) \times \text{Actual Current} \\
 &= \left(\frac{2.5V}{300A} \right) \left(\frac{512}{2.5V} \right) \times \text{Actual Current} \\
 &= (1.7067) \times \text{Actual Current}
 \end{aligned}$$

The C-code for the reading of the current, speed and current reference is given below. The first part is the ADC control code to read the values. The ADC can only read positive values because the input to the ADC is between 0 and 5V. The voltage, representing the current measurement, is offset by 2.5V before being inputted to the ADC. Inside the DSP the zero level is thus at the halve-way value of the 10-bit ADC, which is at 512. The second part is to scale this value back to a 0 reference level, as well as to a 16-bit value because the DSP uses a 16-bit data bus.

```

/* Read data in from ADC6 (Current Ic) and ADC15 (Rotor speed) */

ADCTRL1 -> ADC1CHSEL = 6;          /* Select ADC5 to be read */
ADCTRL1 -> ADC2CHSEL = 7;          /* Select ADC15 to be read */
ADCTRL1 -> ADCSOC = 1;              /* Begin conversion */
while (ADCTRL1 -> ADCINTFLAG == 0); /* Wait for completion of conversion */
ADCTRL1 -> ADCINTFLAG = 1;          /* Reset flag */
IcBack = *ADCFIFO1;                 /* Read in data */
Speed = *ADCFIFO2;

IcBack = IcBack >> 6;                /* Scaling data */
IcBack &= 0x03FF;
IcBack = IcBack - 512;

Speed = Speed >> 6;                 /* Scaling data */
Speed &= 0x03FF;
Speed = -(Speed - 512);              /* From 5V to 2.5V max. */

```


Following this is the reading of the position. This is done using a resolver together with its digital circuitry. This gives the DSP an 11-bit accurate mechanical angle and from that a 10-bit accurate electrical angle of the machine's rotor position or obtained. Thus the angle for 0 to 360° is equal to 0 to 1023 in the DSP, which gives an increment in the DSP of 0.2° mechanical and 0.4° electrical.

The position is read in as follows:

```

/* Reading the position of the rotor */

/* The mechanical rotor angle is 0 - 2047 to give an electrical rotor    */
/* angle of 0 - 1023. The sine table will have a max. magnitude of 255 */

PCDATDIR &= 0xFFFD;          /* Make IOPC1 (FREEZE) = 0 */
PCDATDIR &= 0xFFFE;          /* Make IOPC0 (ENABLE) = 0 */
MrotorAngle = *Datain;
PCDATDIR |= 0x0003;          /* Make IOPC1 (FREEZE) = 1 */
                             /* Make IOPC0 (ENABLE) = 1 */

MrotorAngle = MrotorAngle >> 5; /* Make position a 16-bit value */
MrotorAngle &= 0x07FF;
MrotorAngle = MrotorAngle + 757; /* Zero angle by adding a constant */
if (MrotorAngle > 2047)
    {MrotorAngle = MrotorAngle - 2047;}

ErotorAngle = MrotorAngle;     /* Calculate the electrical rotor angle */
if (ErotorAngle > 1023)
    {ErotorAngle = ErotorAngle - 1023;}

```

The position value is read in from the resolver in the first part and saved into *MrotorAngle*. The resolver is 12-bit accurate and is converted to a 16-bit value for the DSP calculations. A fixed angle is added to the mechanical rotor angle to line-up the zero position of the rotor to the correct position of the stator. The last part of the code is the calculation of the electrical angle, *ErotorAngle*.

After the input of all parameters the measured currents are converted to the *dq0*-reference plane. As mentioned earlier the Park-transformation is used for this and the inverse of this transformation is used at the end of the program to transform the result

of the controllers back to the 3-phase plane to obtain three reference values for the inverter. For these transformations the angles used are in the 0 - 1023 range. The code for the Park-transformation is given below as:

```

/* Calculation of angles for the Park-transformation */

angleA1 = ErotoAngle + 256;          /* Elec. Angle + 90 grade */
if (angleA1 > 1023)
    {angleA1 = angleA1 - 1023;}
angleB1 = ErotoAngle - 341;          /* Elec. Angle - 120 grade */
if (angleB1 < 0)
    {angleB1 = angleB1 + 1023;}
angleB2 = ErotoAngle - 85;           /* Elec. Angle - 120 grade + 90 grade */
if (angleB2 < 0)
    {angleB2 = angleB2 + 1023;}
angleC1 = ErotoAngle + 341;          /* Elec. Angle + 120 grade */
if (angleC1 > 1023)
    {angleC1 = angleC1 - 1023;}
angleC2 = ErotoAngle + 597;          /* Elec. Angle + 120 grade + 90 grade */
if (angleC2 > 1023)
    {angleC2 = angleC2 - 1023;}

```

The transformation and it's inverse is done by first scaling down the input values to prevent overflow in the DSP (see Fig. 4.11). Then the transformation equations are realized. The sine of the angles is obtained from a sine look-up table that is loaded, together with the program, into the DSP. After the transformation, the values are scaled up to their original scale. This scaling up and down of the values influence the accuracy and the dynamics of the current controller. The reason for the scaling is to prevent overflow within the processor. The two transformations in the C-program is:

```

/* abc-to-qdo transformation (Park -1) */

IaBack = IaBack / 4;                /* Scaling */
IbBack = IbBack / 4;
IcBack = IcBack / 4;

IqBack = (IaBack*sine(angleA1))/256 + (IbBack*sine(angleB2))/256 +
(IcBack*sine(angleC2))/256;
IdBack = (IaBack*sine(ErotoAngle))/256 + (IbBack*sine(angleB1))/256 +
(IcBack*sine(angleC1))/256;

IqBack = 4 * (IqBack*2/3);
IdBack = 4 * (IdBack*2/3);

```



```

/* qdo-to-abc transformation (Park) */

Vd = Vd / 16;                      /* Scaling */
Vq = Vq / 16;

Va = (Vd*sine(ErotorAngle))/256 + (Vq*sine(angleA1))/256;
Vb = (Vd*sine(angleB1) )/256 + (Vq*sine(angleB2))/256;
Vc = (Vd*sine(angleC1) )/256 + (Vq*sine(angleC2))/256;

Va = 16*Va;                        /* Scaling */
Vb = 16*Vb;
Vc = 16*Vc;

```

Next is the program code of the two PI controllers. This part of the code makes use of floating-point values because of the accuracy of the constants of the controllers. The code is given as follows:

```

/* D-axis PI Controller */

PidIn = IdRef - IdBack;             /* D-as Current error */

PidOut = PidOutZ + 9.99*(PidIn - PidInZ) + 0.0099*PidIn;
/* U = U(Z-1) + Kpd*(I - I(z-1)) + Kid*T*I */

PidOutZ = PidOut;                  /* Saving data for next loop */
PidInZ = PidIn;
Vd = (int) PidOut;                  /* Convert value to integer */

```

```

/* Q-axis PI Controller */

PlqIn = IqRef - IqBack;            /* Q-as Current error */

PlqOut = PlqOutZ + 4.95*(PlqIn - PlqInZ) + 0.05*PlqIn;
/* U = U(Z-1) + Kpq*(I - I(z-1)) + Kiq*T*I */

PlqOutZ = PlqOut;                  /* Saving data for next loop */
PlqInZ = PlqIn;
Vq = (int) PlqOut;                  /* Convert value to integer */

```


For each controller the current error is calculated were-after the PI transfer function is implemented. The outputs of the transfer functions are saved into two variables to be available for the next loop as previous values. The initial values for these two variables are set at zero.

After the PI controllers, the decoupling of the cross-magnetization and the speed voltages are done. In the case of the cross-magnetization two look-up tables are generated from the results gathered from the finite element analysis as is given by Fig. 2.7. The derivative of the currents is calculated by using the following equation:

$$\frac{di_x}{dt} = \frac{i_x - i_x z^{-1}}{T} \quad (4.17)$$

where $x = d$ or q and T is the sampling period.

The denominator of equation (4.17) is calculated by using the values of the present current and that of the previous sampling period. If the derivative goes negative, the polarity of the cross-magnetization also changes. This is implemented in the second part of the program code. This code is given below. The divider of equation (4.17), which is the sampling period, is incorporated into the look-up tables as shown in the following equation:

$$M_x \frac{di_x}{dt} = \left(\frac{M_x}{T} \right) (i_x - i_x z^{-1})$$

Also included in the look-up tables is a scaling factor to make the values of the look-up tables larger because of the magnitude of the values taking in to account that the processor is a fixed-point processor and can not use values between zero and one. The last part of this code is the addition of these voltages to the output control voltages of the PI controllers.

```

/* Decoupling of Md */

dlq_dt = lqBack - lqBackZ;
lqBackZ = lqBack;
if (ldBack < 0)
    {MdInd = md(-ldBack);
    MdInd = -MdInd;}
else
    {MdInd = md(ldBack);}
MdDec = (MdInd * dlq_dt * -1) / 16;
Vd = Vd + MdDec; /* Decouple the inductance */

```

```

/* Decoupling of Mq */

dld_dt = ldBack - ldBackZ;
ldBackZ = ldBack;
if (lqBack < 0)
    {MqInd = mq(-lqBack);
    MqInd = -MqInd;}
else
    {MqInd = mq(lqBack);}
MqDec = (MqInd * dld_dt * -1) / 16;
Vq = Vq + MqDec; /* Decouple the inductance */

```

Finally the speed-voltages are decoupled. The flux linkages that are used to calculate the speed-voltages are obtained by using look-up tables for the d - and the q -axis. Some of the scaling is also incorporated in the look-up tables. The flux linkages are then multiplied with the electrical speed of the machine. The program code is given below:

```

/* Decoupling of Speed Voltages */

if (ldBack < 0)
    {SpeedVolD = ((-1 * fluxd(-1*ldBack)) * ESpeedBack) / 32;}
else
    {SpeedVolD = (fluxd(ldBack) * ESpeedBack) / 32;}

if (lqBack < 0)
    {SpeedVolQ = ((-1 * fluxq(-1*lqBack)) * ESpeedBack) / 32;}
else
    {SpeedVolQ = (fluxq(lqBack) * ESpeedBack) / 32;}

```


The complete program code for the current controller is given in the appendixes.

4.4 Simulation and testing of the current controller

In this section the two current loops of the controller are simulated separately and are also tested on the actual RSM drive system. The simulated and the measured results are compared to evaluate the model of the controller used in the simulation. Together with this, different scaling values are used in the DSP and the influence of this scaling on the current loop response is evaluated. These tests will be done while the rotor of the machine was fixed in a stationary position.

4.4.1 Simulation of the current loops

A full load step command is given to the d -axis current and the response is simulated. Given below in Fig. 4.12 is the simulated response of the actual (sampled) d -axis current in relation to the reference current. The q -axis current and the speed is kept zero in this simulation and thus do not have any effect on the d -axis response. However the saturation of the d -axis flux does have an influence.

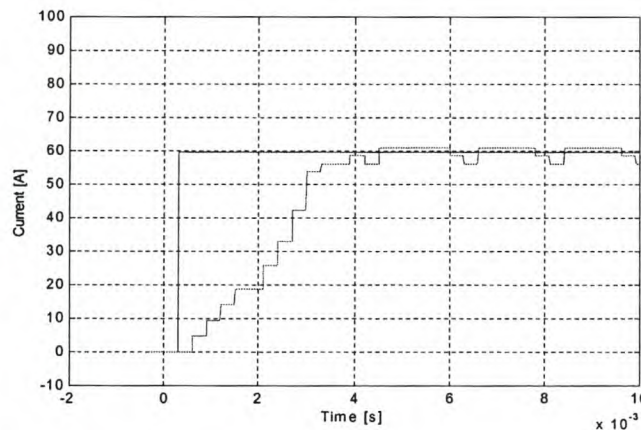


Figure 4.12 – Simulated Full load current step in the d -axis

The settling time is about 3 milliseconds, which is longer than the 900 μ seconds settling time designed for. This is due to the fact that in the design the inductance was kept constant, as where in the simulation the inductance is not constant because of the saturation of the d -axis flux. At low current the d -axis inductance is much larger than at high current.

Another effect on the current response is the truncation that is caused by the fixed-point processing of the controller. This can be seen in the steady state part of the response where the fluctuations in the current are visible. Together with this the way of scaling used, to prevent overflow, is also a cause of the fluctuations. The scaling of the values and the effect of this on the response of the currents are discussed later in this chapter.

A q -axis full load command step is also given, with the d -axis current and speed fixed at zero. The simulated response is given below in Fig. 4.13.

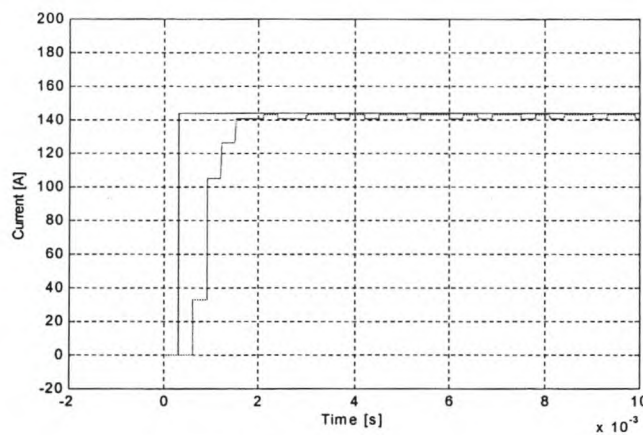


Figure 4.13 – Simulated Full load current step in the q -axis

The settling time is about 1.5 milliseconds and compares well with the designed settling time of 1.21 milliseconds. The designed- and the simulated response times are closer to each other than that for the d -axis because of the much less variation in the q -axis inductance. There are also fluctuations in the steady state part of the current due to the fixed-point calculations and the scaling.

4.4.2 Practical testing of current loops

The results of the previous section must be verified on the actual RSM drive system. The current controllers are tested separately. The measured plots are obtained by writing the values of the reference current and the actual current, as it is inside the digital controller, to the digital-to-analog converters. The values are then displayed as analog signals on an oscilloscope. The noise on the plots is mainly due to switching noise from the drive.

While the q -axis current and the speed is kept at zero, the d -axis command current is given a full-load step. The plot below gives the response of the sampled machine current in this axis.

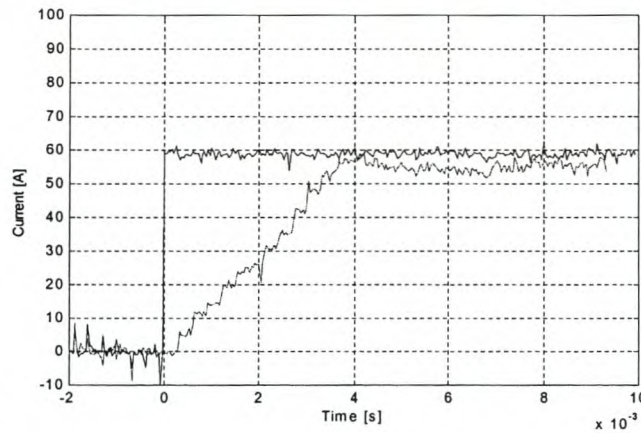


Figure 4.14 – Measured Full-load current step in the d -axis with $I_q = 0A$

From this result one can see that the settling time of the current loop is about 3.5 milliseconds, which is a bit slower than the simulated and designed time, but is still satisfactory. As seen with the simulated results, the difference between the desired and actual currents in the steady state, due to the scaling and truncation of values, is visible in Fig. 4.4. The effect of the saturation can also be seen well.

The q -axis current controller is tested in the same way as the d -axis current controller. The d -axis current and the speed are kept at zero and a full-load step reference is given to the q -axis current loop. The response of the current in the machine is given below in Fig. 4.15.

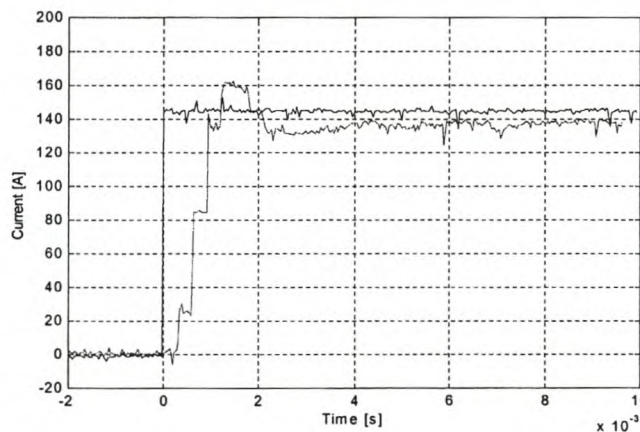


Figure 4.15 – Measured Full-load current step in the q -axis with $I_d = 0A$

This step has a response of about 2 milliseconds, that is also a bit slower than the simulated and designed responses, but it is still within the desired range. The overshoot is a result of the digital controller that has a delayed reaction because of the fact that it can only react to an error only one time every switching cycle.

4.4.3 Scaling effect of fixed point processing

With fixed-point processors the obstacle is to implement an equation by only using integers. No decimal values can be used. This means that the result of every calculation is truncated. The result of this is that values change in integer steps that is not smooth as with real values. The truncation is therefore incorporated into the simulation to simulate the practical system as best as possible.

The second obstacle is the overflow of the processor. The values that are added or multiplied together must be scaled to prevent overflow, that is to prevent the answer being larger than 15-bits (-32768 to 32767). The F240 DSP is a 16-bit processor, but the 16th bit is the positive or negative sign.

Most of the scaling is done in the Park-transformations. The PI controllers are implemented using floating-point processing because of the magnitudes of the proportional and integral constants. This floating-point processing inside the fixed-point processor takes a large time to execute and is therefore limited to be used only for the PI controllers. Two types of scaling in the Park-transformation are used and the difference in the control performance between these types is evaluated.

For the first scaling type the sine table that is used to get the sine of the rotor angle is set up as shown in Fig. 4.16.

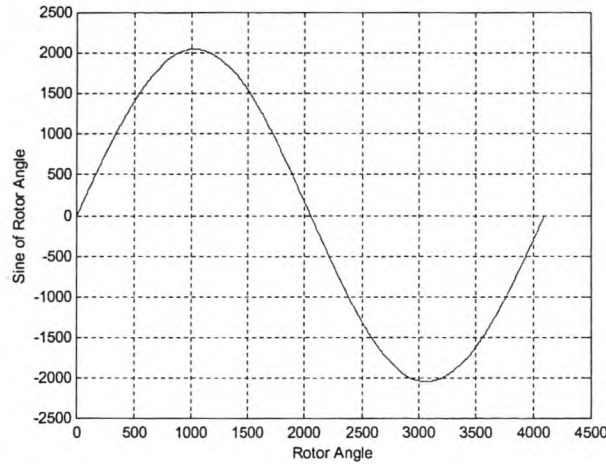


Figure 4.16 - 12-bit sine waveform of the lookup-table used

In this table an angle between 0 and 360° is represented in the DSP as between 0 and 4095 with the y-axis values between 2047 and -2048. These values fix the maximum size of the sine of an angle. Looking at the equations of the inverse Park-transformation, a sine of an angle is multiplied by a measured- or reference current value or voltage value. In the program code below $I_{a.back}$, $I_{b.back}$ and $I_{c.back}$ are the values of the phase currents read in of the machine. Due to the 10-bit ADC's, these values varies from +511 to -512. The product of the maximum values of one of the currents and that of the sine lookup-table is 1.046×10^6 . This value is greater than the maximum value the processor can handle. For this reason both the values of the phase currents and the sine lookup-table are scaled down. The sine of the angle is scaled by a factor 4 to a maximum value of 512 and the current by a factor of 8 to a maximum value of 64. The product of these two values is then 32,768. This value will not cause an overflow in the processor. After the calculations the currents is scaled up by the factor of 8 so that the input currents to the transformation is in relation with the output current values. The following is the program code of the inverse Park-transformation with the scaling of the currents and sine-table.

```

laTerug = laTerug / 8;
lbTerug = lbTerug / 8;
lcTerug = lcTerug / 8;

IqTerug = (laTerug*(sine(hoekA1) / 4)) / 512 + (lbTerug*(sine(hoekB2) / 4)) / 512 +
(lcTerug*(sine(hoekC2) / 4)) / 512;
IdTerug = (laTerug*(sine(Erotorhoek) / 4)) / 512 + (lbTerug*(sine(hoekB1) / 4)) / 512 +
(lcTerug*(sine(hoekC1) / 4)) / 512;

IqTerug = 8 * (IqTerug * 2/3);
IdTerug = 8 * (IdTerug * 2/3);

```

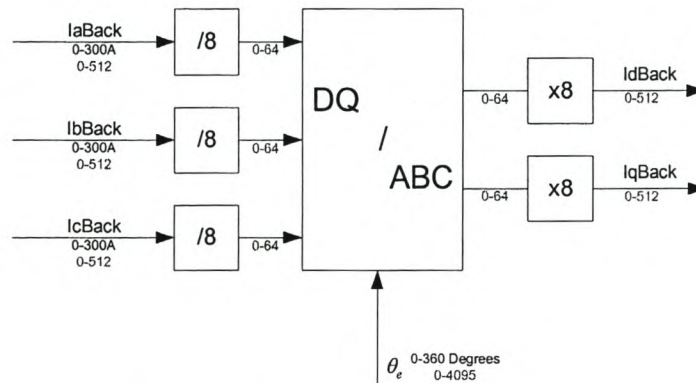


Figure 4.17 - First type of scaling of values in inverse Park-transformation

The bad side of scaling the sine of the angle and the values of the currents or voltages is that a lot of resolution is lost in these values due to truncation.

The second type of scaling is an improved scaling method. The values of the sine-table are decreased by using only 10-bit resolution on the rotor angle. This results in a maximum sine value of 255 in relation with the previous value of 2048. The sine waveform of the lookup-table is given below in Fig. 4.18.

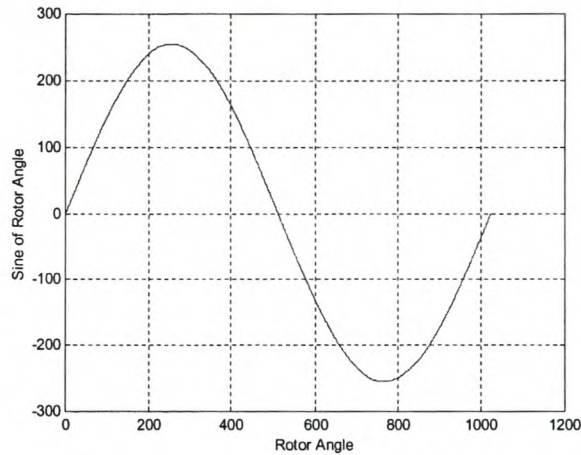


Figure 4.18 - 10-bit sine waveform of the lookup-table used

The lower maximum value of the sine of an angle in this case means that it is not necessary to scale this value again. The accuracy of the angle is also not affected in a large way, because with a 10-bit resolution every 0.703 degrees gives 1 bit. Furthermore, the current values do not have to be scaled as much as before. They are scaled by a factor 4 from a maximum of 512 from the DAC's to a maximum of 128 (see Fig. 4.19). This is double the resolution of the previous scaling method. The code for this in the processor is:

```

IaBack = IaBack / 4;
IbBack = IbBack / 4;
IcBack = IcBack / 4;

IqBack = (IaBack*sine(angleA1) )/256 + (IbBack*sine(angleB2))/256 +
(IcBack*sine(angleC2))/256;
IdBack    =    (IaBack*sine(ErotorAngle))/256    +    (IbBack*sine(angleB1))/256    +
(IcBack*sine(angleC1))/256;

IqBack = 4 * (IqBack*2/3);
IdBack = 4 * (IdBack*2/3);

```

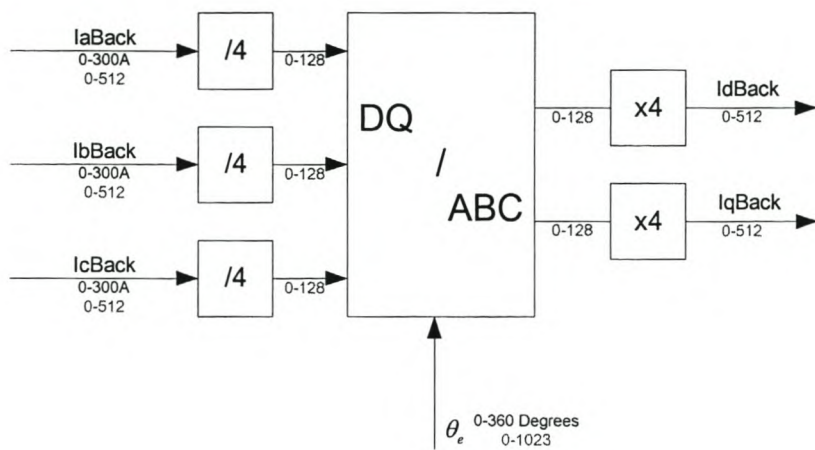



Figure 4.19 - Improved scaling of the Park-transformation

Both these scaling methods are implemented in the simulation model. The difference in response between the two is evaluated with a step in the d -axis current. Simulated results are shown in Figs. 4.20 and 4.21 and measured results in Figs. 4.22 and 4.23. The methods are also implemented on the actual drive system to see the effect thereof.

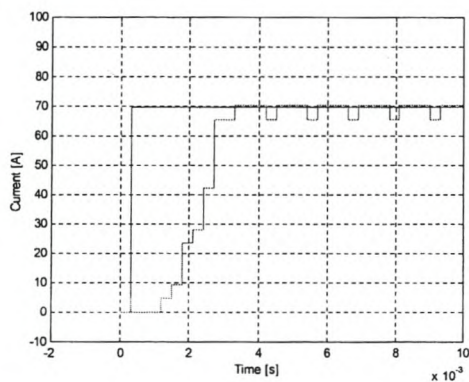


Figure 4.20 – Simulated d -axis response with the first scaling method

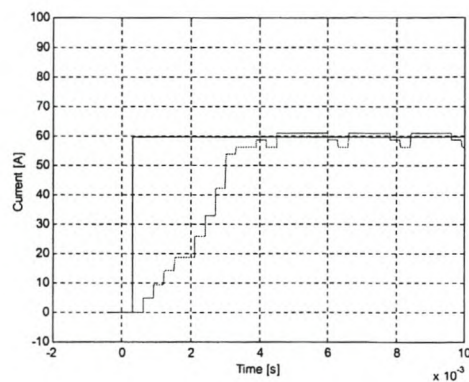


Figure 4.21 – Simulated d -axis response with the improved scaling method

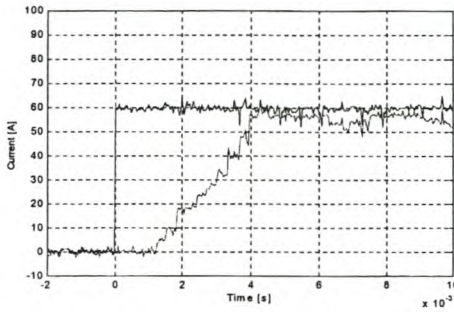


Figure 4.22 – Measured d -axis response with the first scaling method

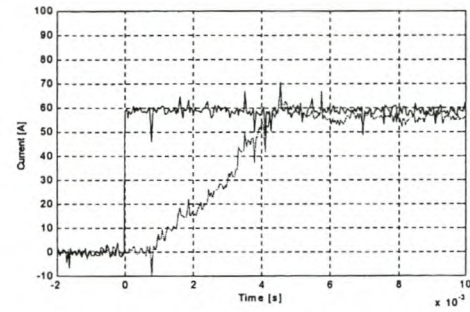


Figure 4.23 – Measured d -axis response with the improved scaling method

As seen in Figs 4.20 and 4.21 it is apparent that there is a difference in response between these scaling methods. From these results it can be seen that with the first scaling method the steps of the current is much larger than in the case of the improved scaling method. The second method's results are also more stable; there is less oscillation than with the first one. These results conclude that great care must be taken with the scaling of fixed-point values. The dividing of a parameter for scaling down causes a downscaling in the resolution of the value, because of the truncation effect in the fixed-point processor.

4.4.4 Decoupling of cross-magnetization

The effect of decoupling of the cross-magnetization terms on the response of the current controller is evaluated in this section. The evaluation is done by keeping the d -axis current constant and giving then a full-load step of the q -axis current. This step gives a disturbance in the d -axis current due to the mutual coupling between the d - and q -axis circuits. Shown in Fig. 4.24 are the simulated (green) and the measured (blue) result of the q -axis cross-magnetization voltage.

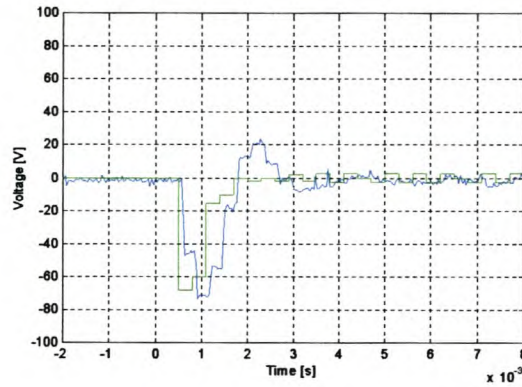


Figure 4.24 - d -axis cross-magnetization voltage with change in q -axis current.

The simulated and the measured results are not much different and it can be seen that a maximum voltage of around 70V for this disturbance is generated.

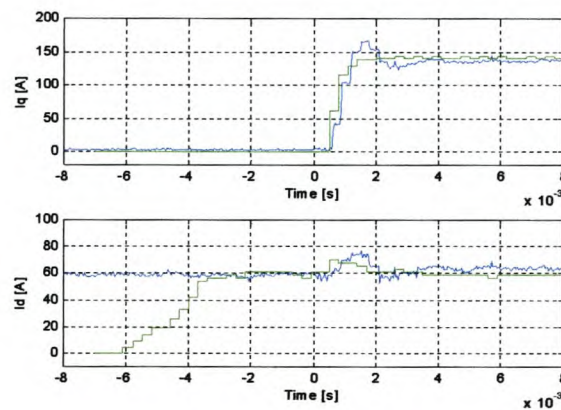


Figure 4.25 - d -axis current with step in q -axis current without decoupling

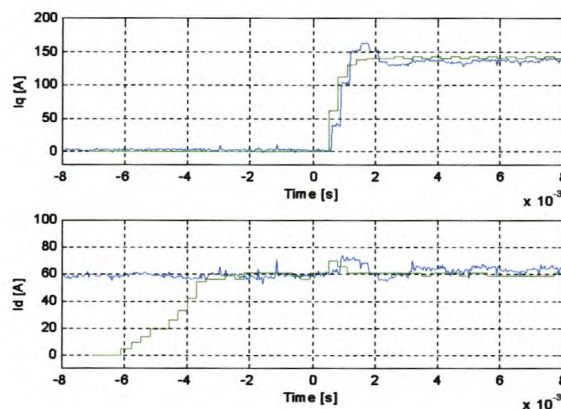


Figure 4.26 - d -axis current with step in q -axis current with decoupling

Fig. 4.25 shown the influence of the cross-magnetization on the simulated (green) and measured (blue) i_d response with a step in the i_q . With the decoupling the response of i_d with a step in i_q is shown in Fig. 4.26. With decoupling the response is faster than

without the decoupling. The measured results show that the difference is less between the two cases, but there is an improvement in the magnitude of the disturbance. The cross-magnetization component is unfortunately not completely removed because of the value of this component being an estimation of the decoupling within the controller and not exact component. The test was done at standstill, but the influence of the cross-magnetization must also be investigated at rated speed or higher.

4.5 Current control methods

The previous sections described the design, simulation and implementation of the current controller. The current controller has two reference inputs, namely the d -axis and the q -axis reference current inputs. There are two methods of controlling these two inputs. The conventional control method (also in general for drives) is to keep the d -axis current constant (fixed field) below the rated speed, and in the high-speed region to reduce the d -axis current going into field weakening. Another control method is to keep the current angle constant. With this method the magnitude of the current vector is used and does not have a constant d -axis current flowing. This is a more energy efficient method than the conventional control method.

4.5.1 Constant d -axis current control method (Conventional Method)

The name of the method explains the strategy. The d -axis current is held constant while the torque is controlled by the q -axis current control. The constant d -axis current ensures a constant d -axis flux in the machine. The disadvantage of this method is that the d -axis current is flowing at no-load which is from an energy-efficient point of view not good. The evaluation of this method is done in chapter 7.

4.5.2 Constant current angle control method

This method keeps the current angle ϕ of the stator current vector constant. The current vector \bar{I}_s is given in Fig. 2.9. The torque is controlled by controlling the magnitude of \bar{I}_s . The advantage of this method is that the current in both the d -axis and q -axis is at a minimum at no-load. This is good from an energy-efficient point.

For the implementation of the constant current angle current control the equation for the stator current vector can be divided into its d - and q -axis components. This is given as

$$i_d = I_s \cos(\phi) \quad (4.18)$$

$$i_q = I_s \sin(\phi) \quad (4.19)$$

where I_s is the magnitude and ϕ the angle of the current vector as shown in Fig. 2.9. These equations are implemented in the simulation as given in Fig. 4.27 and the same scaling is used as with the Park-transformation. The evaluation of the method is discussed in chapter 7.

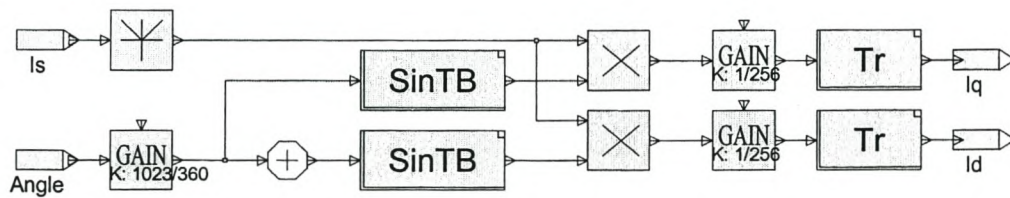


Figure 4.27 - Simulation block diagram of the transformation of the current vector to dq-currents

5 - Digital speed-control of the RSM

This Chapter's aim is the development of a speed controller for the RSM drive. The control can be done with the conventional constant d -axis current controller or with the constant current angle controller. Firstly the constant- d -axis-current speed controller is designed and evaluated; thereafter the constant-current-angle speed controller is evaluated in detail. The two controllers are compared to see the advantages and disadvantages of the constant-current-angle speed controller.

5.1 Constant- d -axis-current speed control

In this section the design and evaluation of the constant- d -axis-current speed controller is done. In the previous chapter the current controller for this type of current control was briefly described. The first part of this section explains the relation between the current and the torque produced of this type of current control. Thereafter the transfer function of the speed control loop is dealt with and the design of the PI controller is done. Finally the controller is simulated and tested on the actual drive system.

5.1.1 Constant- d -axis-current controller

The relation between the current and the developed torque for this current control method can be described by taking a look at the torque equation of the machine as it is given by equation (2.20). The flux linkages λ_d and λ_q of equation (2.20) can, as an approximation, be written in terms of constant inductances as

$$\lambda_q = L_q i_q \text{ and } \lambda_d = L_d i_d \quad (5.1)$$

The torque equation (eqn. 2.20) can be rewritten as

$$T_{em} = \frac{3}{2} p (L_d - L_q) i_d i_q. \quad (5.2)$$

With the d -axis current i_d constant, and by assuming as an approximation that L_d' and L_q' are also constant, then equation (5.2) can further be simplified as

$$T_{em} = K_{cc} i_q. \quad (5.3)$$

The steady state measurement of the q -axis current and the developed torque is done for different i_q -current values. These results are given in Table 5.1 below.

$I_{q,ref}[A]$	$I_{q,ref} * 512/300 [DSP]$	$T_{em} [Nm]$	K_{CC}
144	245	296	1.21
120	204	251	1.23
100	170	206	1.21
80	136	167	1.23
60	102	124	1.22
40	68	82	1.21
20	34	38	1.12
0	0	0	0

Table 5.1 - Gain of Constant d -axis Current Controller

From these results the gain of the current controller is taken as $K_{cc} = 1.22$.

Equation (5.3) shows that the torque of the system can be controlled by controlling the q -axis current i_q .

5.1.2 Transfer function of the speed control system

The transfer function of the speed controller consists of a few components. It consists of the current controller, the mechanical system, the speed controller and scaling constants. The complete block diagram of the speed control system is shown in Fig. 5.1.

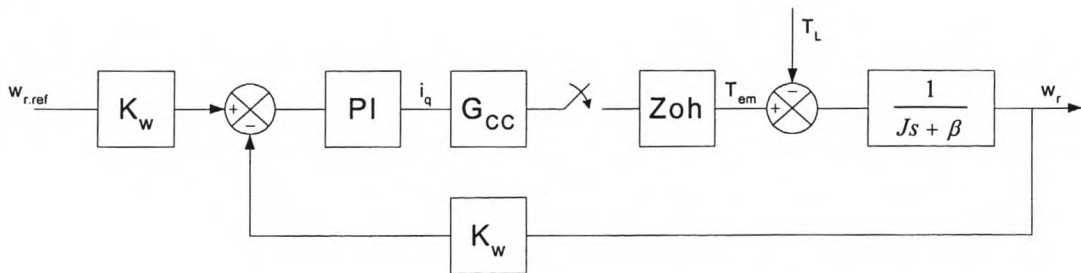


Figure 5.1 – Block diagram of speed control loop

The transfer function between the reference current i_q^* and the developed torque T_{em} for the digital current controller is:

$$G_{cc}(s) = \left(\frac{T_{em}}{i_s} \right) \left(\frac{i_s}{i_s^*} \right) = K_{cc} \left(\frac{1}{\tau_{cc}s + 1} \right) \quad (5.4)$$

where τ_{cc} is the time constant of the current controller and K_{cc} the gain of the transfer function. The time constant depends on the settling time of the q -axis current, which will be zero for an ideal current controller. To determine τ_{cc} and K_{cc} a full-load step is given to the q -axis current controller with the d -axis current held constant at its rated value of 60A. The response of the q -axis current is obtained from the simulation and the actual RSM drive system and is given in Fig. 5.2.

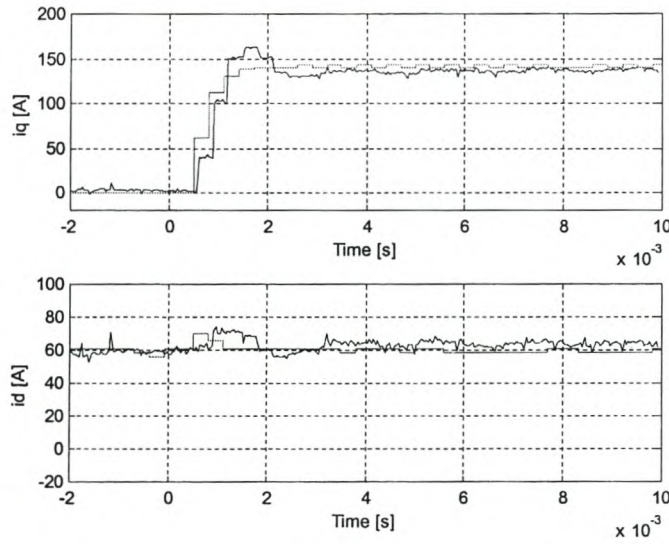


Figure 5.2 – Measured and Simulated q -axis current step response with i_d constant.

From this result it can be seen that the settling time of the q -axis current is in the region of 2 milliseconds with the d -axis current constant.

Fig. 5.3 shows the q -axis current together with the developed torque response of the machine.

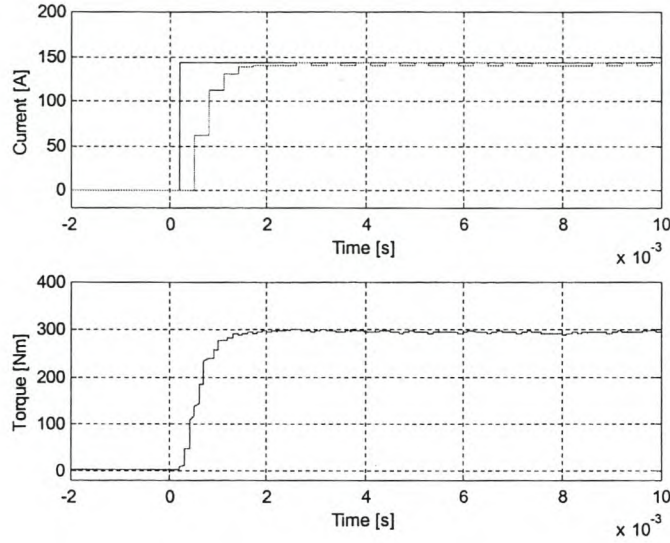


Figure 5.3 – Simulated q -axis current and developed torque step response with constant d -axis current.

The transfer function between the q -axis reference current and the developed torque is:

$$G_{cc}(s) = \frac{1.22}{0.002s + 1} \quad (5.5)$$

This equation can be simplified by ignoring the time constant because of the fact that the response of the current controller is much faster than that of the speed loop. The transfer function is thus simplified to:

$$G_{cc}(s) = 1.22 \quad (5.6)$$

The next component is the PI speed controller and its transfer function is similar than the PI current control transfer function given by equation (4.6). This transfer function is in the z -plane because of the digital controller. This means that the Zoh-circuit is also present like in the transfer function of the two current controllers. The transfer function of the Zoh-circuit is given by equation (4.7).

The last component is the transfer function of the mechanical system. This consists of the inertia and the friction of the electrical machine and load system. The transfer function can be written as follows:

$$G(s) = \frac{1}{Js + \beta} \quad (5.7)$$

where J is the inertia and β the friction.

All of these components' transfer functions must be transformed to the z -plane for the design of the digital controller. The transfer function of the PI controller is already expressed in the z -plane and the constants are the same in the s -plane as in the z -plane.

The total block diagram of the system can be simplified to the following diagram. The influence of the load torque is ignored in the design of the speed controller as the PI speed controller is designed for good response on the speed step input.

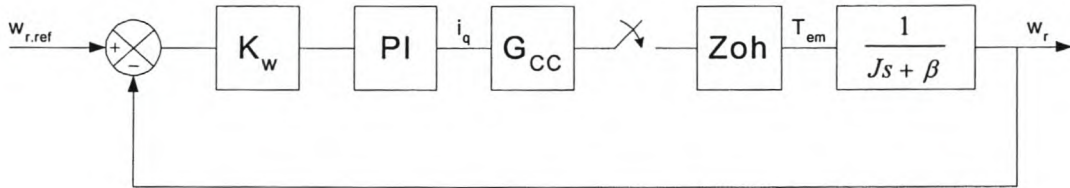


Figure 5.4 - Speed loop

The transfer function of the mechanical system is given by eqn. (5.6) in the s -plane. This equation together with that of the Zoh-circuit is transformed into the z -plane. These two are transformed together in one transfer function and is given by:

$$G(z) = \frac{1 - e^{-\frac{\beta T}{J}}}{\beta \left(z - e^{-\frac{\beta T}{J}} \right)} \quad (5.8)$$

From Equations (4.6), (5.5) and (5.7), the total transfer function of the speed control system is determined as follows:

$$G_{cl}(z) = \frac{\frac{K_{cc}K_w}{\beta} \left[1 - e^{-\frac{\beta T}{J}} \right] \left[(K_p + TK_i)z - K_p \right]}{z^2 + \left[\left(-1 - e^{-\frac{\beta T}{J}} \right) + \frac{K_{cc}K_w}{\beta} (K_p + TK_i) \left(1 - e^{-\frac{\beta T}{J}} \right) \right] z + \left[e^{-\frac{\beta T}{J}} - \frac{K_p K_{cc} K_w}{\beta} \left(1 - e^{-\frac{\beta T}{J}} \right) \right]} \quad (5.9)$$

This equation is used in the design of the PI speed controller or in particular the proportional- and integral constants: K_p and K_i .

5.1.3 Root locus design of the PI speed controller

In the previous section the closed loop transfer function of the digital speed control is derived. The values of the constants are:

$J = 0.8 \text{ kg.m}^2$	(Inertia of rotor)
$\beta = 0.1$	(Friction)
$K_w = 512/300 = 1.7067$	(Scaling constant for speed measurement)
$K_{cc} = 1.22$	(Transfer function of inverter)
$T = 300\mu\text{S}$	(Sampling period)

The open loop transfer function of the system is calculated by substituting the values above into eqn. (5.8) and with K_{cc} and is given as:

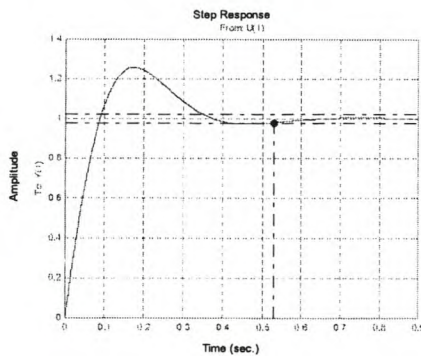
$$G_{ol}(z) = D(z) \frac{767.986e^{-6}}{z - 0.99996} \quad (5.10)$$

Using this transfer function a root locus design is done to determine the transfer function of the PI controller. This design is done with the help of MATLAB and the specifications of this controller is:

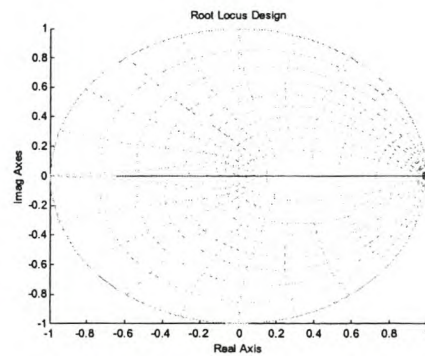
$$\tau_s \leq 500 \text{ milliseconds}$$

$$\xi < 1 \text{ (Under damped)}$$

The root locus is shown in Fig. 5.5(b) and the unit step response of the system in Fig. 5.5(a).



(a)



(b)

Figure 5.5 - (a) Step Response and (b) Root-locus of the speed loop

The system is under damped with a minimum overshoot. The settling time is in the region of 500 milliseconds. From the root locus the transfer function for the PI speed controller is:

$$D(z) = \frac{6.021(z - 0.9965)}{z - 1}$$

$$= \frac{6.021z - 6}{z - 1} \quad (5.11)$$

where the proportional and integral constants are: $K_p = 6$ and $K_i = 70$.

5.2 Constant-current-angle speed control

This section describes at the design, implementation and evaluation of the constant-current-angle speed control technique. In this controller the magnitude of the current vector is controlled while the vector's angle is kept constant.

5.2.1 Constant-current-angle speed controller

The torque of the reluctance synchronous machine, given in equation (2.20), is used to describe the constant-current-angle speed control method. By using the same relation of the flux linkage given by equation (5.1) and the relation between the dq -axis currents and the current vector,

$$i_d = i_s \cos \phi \text{ and } i_q = i_s \sin \phi, \quad (5.12)$$

the torque equation can be rewritten as:

$$T_{em} = \frac{3}{4} P (L'_d - L'_q) i_s^2 \sin(2\phi). \quad (5.13)$$

From this equation the developed torque of the machine is proportional to the square of the magnitude of the current vector. Thus a step in the current vector magnitude keeping the current angle constant, will give a step in the torque of the machine. In the next sections the speed control transfer function is discussed as well as the design of the controller.

5.2.2 Transfer function of the speed control system

The transfer function of the constant current angle speed controller is not much different than that of the constant- d -axis-current speed controller that was discussed in section 5.1.2. The only difference is the transfer function of the current-to-torque conversion. The complete block diagram of the speed control system is shown in Fig. 5.6.

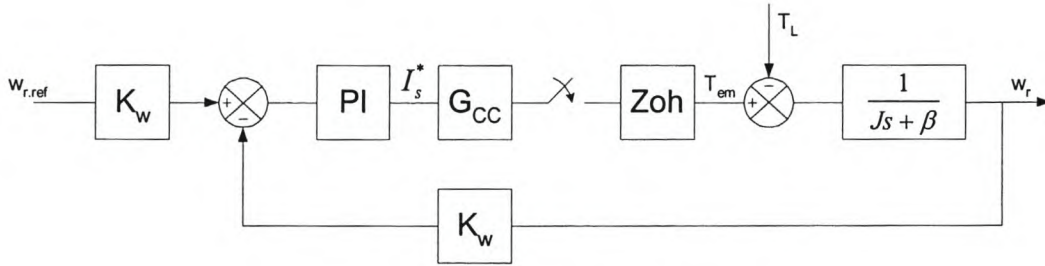


Figure 5.6 – Block diagram of the speed control loop

The transfer function of the current controller is simplified to a first order transfer function as the transition responds is close to a first order responds. The transfer function of the current controller is

$$G_{cc}(s) = K_{cc} \left(\frac{1}{\tau_{cc}s + 1} \right), \quad (5.14)$$

where τ_{cc} is the time constant and K_{cc} the gain of the transfer function. The time constant depends on the settling time of the d -axis and q -axis current or else the I_s current. To obtain the settling time of this transfer function the angle of the current vector is kept constant and a step in the magnitude of I_s is given from zero to full-load current. The measured d - and q -axis currents are given in Fig. 5.7.

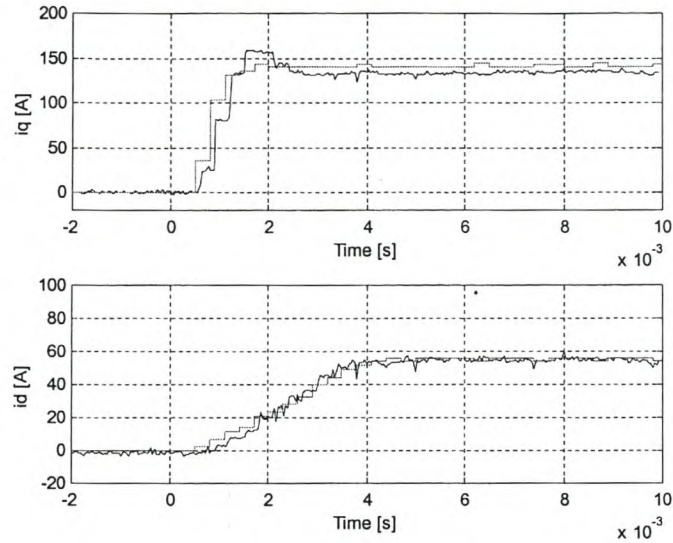


Figure 5.7 - Step in the magnitude of the current vector with a current angle of 68° (measured and simulated dq -currents)

Both the d - and q -axis current loops have different settling times. From the results it can be seen that the settling time of the d -axis current loop is longer than that of the q -axis current. The settling time of the d -axis current is in the region of 3 to 4 milliseconds and this is then also the settling time of the entire current controller. The simulated and measured magnitude of the current vector is given in Fig. 5.8.

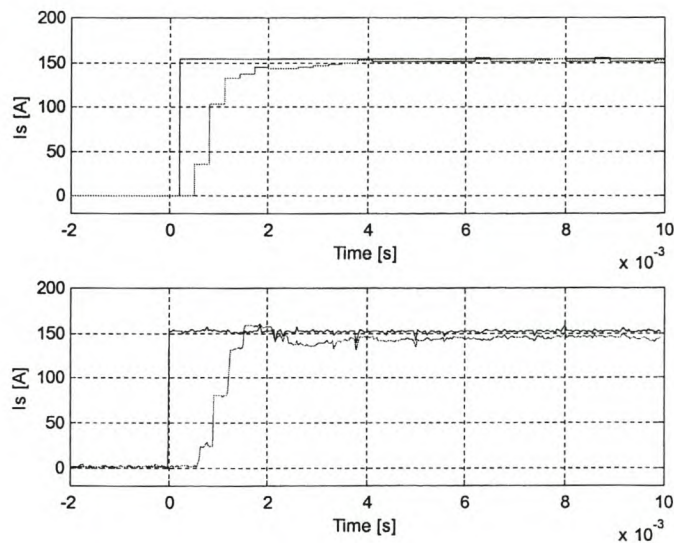


Figure 5.8 - Step in the magnitude of the current vector with a current angle of 68°

The settling time is the same as that of the d -axis current in Fig. 5.7, but the rise time is much faster, because the q -axis current response is faster and more than double the magnitude of the d -axis current.

The gain of the current controller between the reference current and the torque can be calculated by making use of the simulation model. The steady state measurement of the reference current I_s and the developed torque T_{em} is done for different I_s values. These results are given in Table 5.2 below.

$I_{s,ref} [A]$	$I_{s,ref} * 512/300 [DSP]$	$T_{em} [Nm]$	K_{CC}
156	266	295	1.11
140	238	260	1.09
120	204	219	1.07
100	170	180	1.06
80	136	125	0.92
60	102	82	0.80
40	68	33	0.49
20	34	8	0.24
0	0	0	0

Table 5.2 - Gain of constant current angle current controller

From these results the gain of the current controller is averaged to $K_{cc} = 0.93$. Thus the transfer function between the reference current and the developed torque is:

$$G_{cc}(s) = \frac{0.930}{0.004s + 1} \quad (5.15)$$

The time constant of this transfer function is much faster than the time constant of the speed loop and because of this it will have a minimal influence on the speed controller's response. The transfer function for the current controller is then simplified to the following function:

$$G_{cc}(s) \approx 0.930 \quad (5.16)$$

The other components of the speed control loop have been discussed in section 5.1.2. The block diagram of the system, with the influence of the load torque ignored, is given in Fig. 5.9.

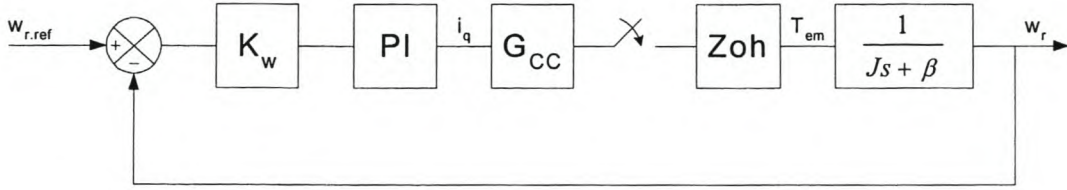


Figure 5.9 - Simplified Constant current angle Speed Control

The closed loop transfer function of the speed control system is described as follows:

$$G_{cl}(z) = \frac{\frac{K_{cc}K_w}{\beta} \left[1 - e^{-\frac{\beta T}{J}} \right] \left[(K_p + TK_i)z - K_p \right]}{z^2 + \left[\left(-1 - e^{-\frac{\beta T}{J}} \right) + \frac{K_{cc}K_w}{\beta} (K_p + TK_i) \left(1 - e^{-\frac{\beta T}{J}} \right) \right] z + \left[e^{-\frac{\beta T}{J}} - \frac{K_p K_{cc} K_w}{\beta} \left(1 - e^{-\frac{\beta T}{J}} \right) \right]} \quad (5.17)$$

This equation is used for the design of the PI speed controller and in particular the proportional and integral constants, K_p and K_i .

5.2.3 Root Locus Design of the PI speed controller

In the previous section the closed loop transfer function of the digital speed control is derived. The values of the constants are:

$J = 0.8 \text{ kg.m}^2$	(Inertia of rotor)
$\beta = 0.1$	(Friction)
$K_w = 512/300 = 1.7067$	(Scaling constant for speed measurement)
$K_{cc} = 0.930$	(Transfer function of inverter)
$T = 300\mu\text{s}$	(Sampling period)

The open loop transfer function of the system is determined by substituting the values given above into eqn. (5.8) and with K_{cc} is given as

$$G_{ol}(z) = D(z) \frac{595.2e^{-6}}{z - 0.99996} \quad (5.18)$$

Using this transfer function a root locus design is done to determine the constants of the transfer function of the PI controller. This design is done with the help of MATLAB and the specifications of this controller is:

$$\begin{aligned} \tau_s &\leq 500 \text{ milliseconds} \\ \xi &< 1 \text{ (Under damped)} \end{aligned}$$

The root locus is shown in Fig. 5.10(b) and the unit step response of the system in Fig. 5.10(a).

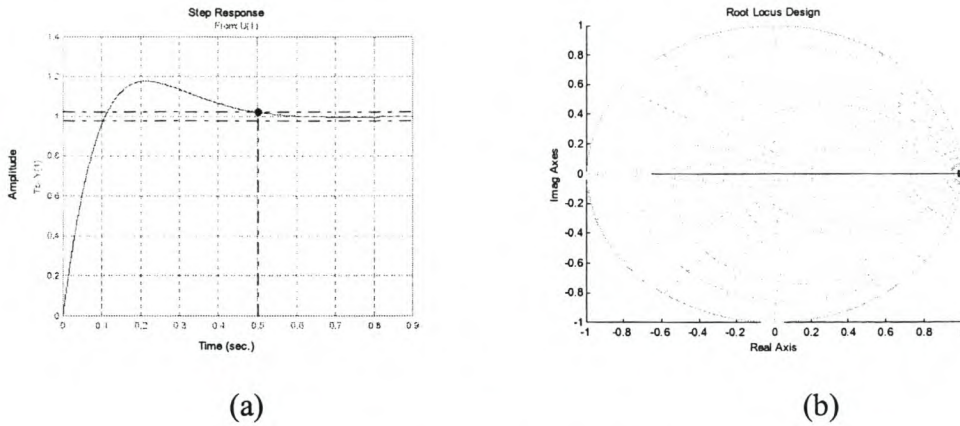


Figure 5.10 - (a) Step Response and (b) Root-locus of the speed loop

The system is under damped with a minimum overshoot. The settling time is in the region of 500 milliseconds, the same as with the design of the constant- d -axis speed controller. From the root locus the transfer function of the PI speed controller is:

$$\begin{aligned}
 D(z) &= \frac{6.0009(z - 0.9985)}{z - 1} \\
 &= \frac{6.009z - 6}{z - 1} \quad (5.19)
 \end{aligned}$$

where the proportional and integral constants are calculated as, $K_p = 6$ and $K_i = 30$.

5.3 Implementation of the speed controller

The speed controller for both of the control methods is implemented in the same way. The speed is calculated from the position measurement, which is more accurate than the measured speed from the resolver circuit. The mechanical rotor position (11-bit resolution) is digitally differentiated to obtain the mechanical speed. The ANSI C-code for the speed calculation is given as:

```

/* Calculating the speed of the rotor */

if ((MrotorAngleZ > 1500) & (MrotorAngle < 500))
    {MrotorAngle = MrotorAngle + 2047;}
if ((MrotorAngleZ < 500) & (MrotorAngle > 1500))
    {MrotorAngleZ = MrotorAngleZ + 2047;}

SpeedBack = 10 * (MrotorAngle - MrotorAngleZ);    /* Convert position to speed */

if (MrotorAngle > 2047)
    {MrotorAngle = MrotorAngle - 2047;}
if (MrotorAngleZ > 2047)
    {MrotorAngleZ = MrotorAngleZ - 2047;}
MrotorAngleZ = MrotorAngle;

ESpeedBack = (32 * SpeedBack) / 19;                /* Scaling of Speed for decoupling */

```

The speed controller is a PI controller, but the integration part of the controller is halted if the output of the controller is limited. This limiting is to limit the current and the torque in the machine. The program code is given as:

```

/* Speed Controller */

PISpeed = SpeedRef - SpeedBack;    /* Calculating the error */
if (Wait < 6000)
    {Wait = Wait + 1;
    PISpeed = 0;}

PISpeedIn = PISpeed;

if ((IsRefZ >= limit) || (IsRefZ <= -limit)) /* Stop integrator in limit */
    {PISpeedIn = 0;}

PISpeedOut = PISpeedOutZ + 6*(PISpeedIn - PISpeedInZ) + 0.009*PISpeedIn;
/* U = U(Z-1) + Kp*[I - I(Z-1)] + Ki*T*I */
PISpeedInZ = PISpeedIn;
PISpeedOutZ = PISpeedOut;
IsRef = (int) PISpeedOut;

```



```
if ((IsRefZ >= limit) || (IsRef <= -limit))
    {IsRef = IsRef + 6*PISpeed ;}
IsRefZ = IsRef;

if (IsRef >= limit)                /* Limit output */
    {IsRef = limit;}
if (IsRef <= -limit)
    {IsRef = -limit;}
```

The complete control programs for the controllers are given in the appendixes.

6 - Hardware of the Digital Controlled RSM Drive

The different parts of the RSM drive are discussed in this chapter. The hardware setup for the drive is shown in Fig. 6.1. The system is divided into four basic parts. These parts are the Supply and Rectifier, Inverter, Controller and Electrical Machine (RSM). Some of the parts were already built and available but others were specifically built for the project and these are discussed in more detail. The drive is fed from an AC supply which is rectified to give the inverter a constant dc-bus voltage. The inverter with the digital controller then generates the three-phase voltages for the machine.

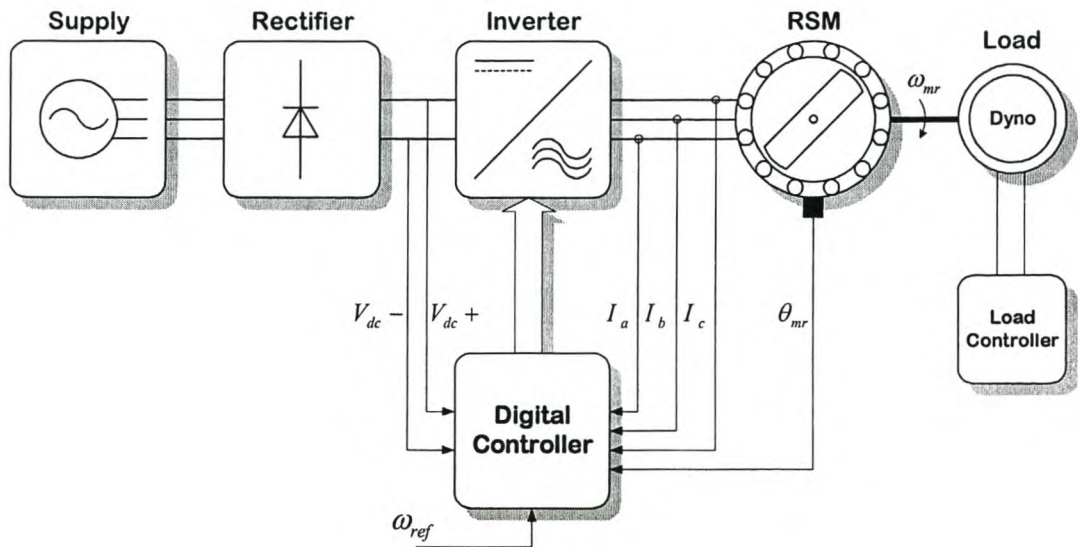


Figure 6.1 - Complete RSM drive

6.1 Supply and Rectifier

The supply to the drive system is a 400V line-to-line AC voltage supply. This supply is able to give up to 400A of line current. The Rectifier is a normal 3-phase AC-to-DC passive rectifier (see Fig. 6.2) that is rated for 200kVA. It rectifies the AC voltage to 550V DC. This DC bus voltage is fed through to the capacitor bank and the inverter.

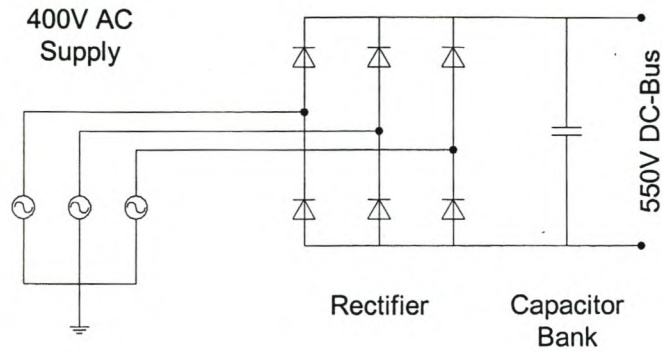


Figure 6.2 - Supply and Rectifier circuit

6.2 Inverter

The inverter used is a 200kVA full-bridge 3-phase IGBT inverter as shown Fig. 6.3. Each phase winding is connected to two phase-arms; each rated at 1200V and 300A. Each phase winding is thus controlled independently. The inverter is built in a unit together with the rectifier. The rectifier/inverter unit also has a full protection circuit, in the case of over current and over temperature faults with the switching devices, as well as a 10kW dump circuit to prevent large over-voltage on the dc-link in the case of regeneration.

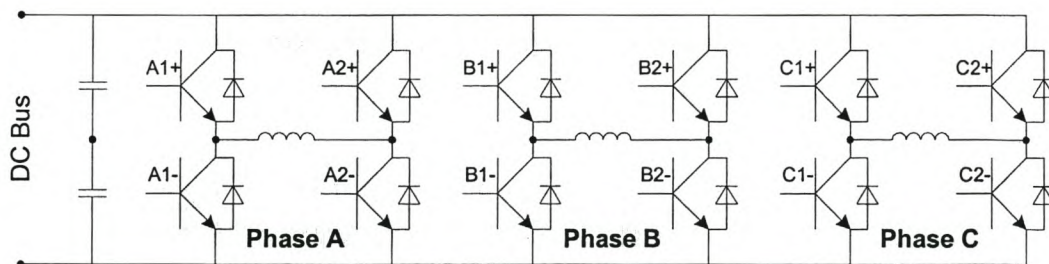


Figure 6.3 - 3 phase full-bridge inverter

The complete rectifier/inverter unit with protection and dump circuit is shown below in Fig. 6.4.

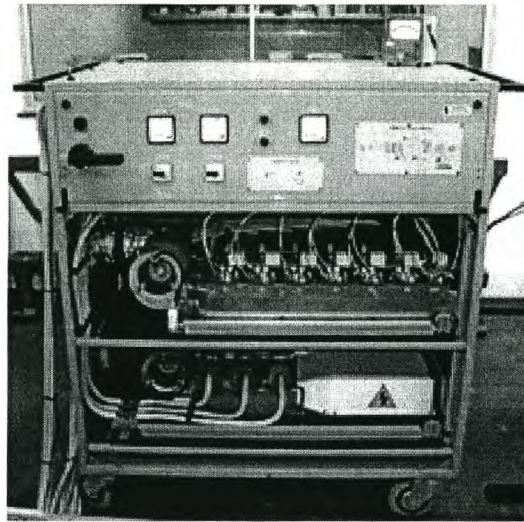


Figure 6.4 - Power Electronic Inverter and Rectifier

6.3 Electrical Machine

The electrical machine is a four-pole 42kW RSM traction machine. It is designed for a maximum line voltage of 500V rms and an output torque of around 296 Nm at full-load.

The stator consists of a chorded 3-phase winding and a core with 48 slots. The rotor is transverse laminated and it is unskewed. The stator and rotor structures were optimum-designed using a finite element optimisation program and is shown in the Fig. 6.5 below.

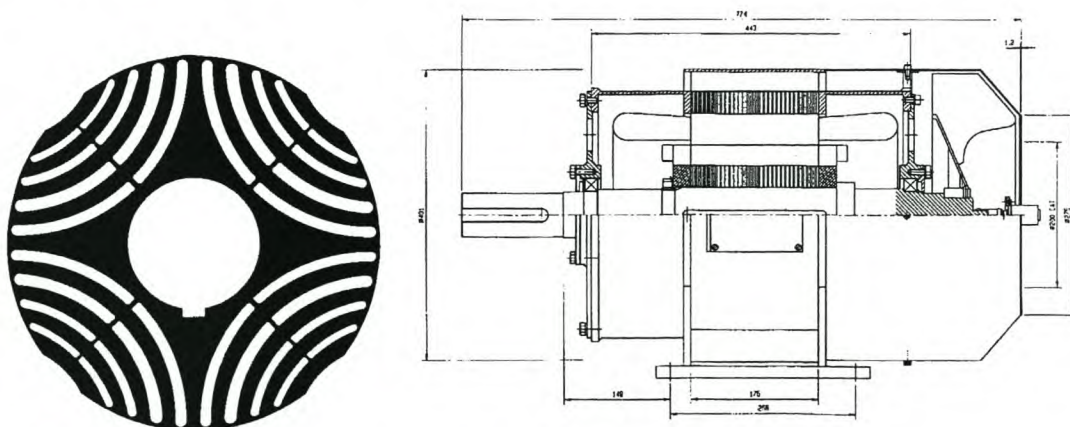


Figure 6.5 - Rotor structure and Motor drawing

6.4 Digital Controller

Nowadays there are a whole range of Digital Signal Processors available. The DSP's are divided into two categories. The first is the floating-point processors. These are normally used for signal analysis and for more advanced applications like in cell phones. The second category is the fixed-point processors. Within this category there are DSP's for specific use, like for motor drive controllers. These DSP's have build-in analog-to-digital converters, PWM outputs and much more for specific use.

The DSP that is chosen for the RSM drive system is part of the TMS320C24x Series. This series is specifically designed for the control of variable speed drives as well as power electronic inverters. The TMS320F240 DSP, which is used, has a few features, namely:

- 16K Flash RAM
- 12 Compare/PWM Channels
- Three 16-Bit General-Purpose Timers
- Dual 10-Bit Analog-to-digital conversion Module
- Serial Communications Interface Module
- Serial Peripheral Interface Module
- 6 External Interrupts (incl. Power Drive Protect interrupt)
- 50ns Instruction Cycle Time

The control unit consists of four blocks as shown in Fig. 6.6. The first and main block is the Processor/Protection block, which hosts the DSP processor. The other three blocks is the fibre optic interface with the inverter drive via fibre optic cable, the rotor position measurement interface, and the current and voltage measurement together with their protection circuitry. They are discussed in the following sections. The blocks is build, each on its own printed circuit board as well as a separate board for the power supply unit. All the circuit boards are fitted nicely into a Euro-rack configuration cabinet (see Fig. 6.7).

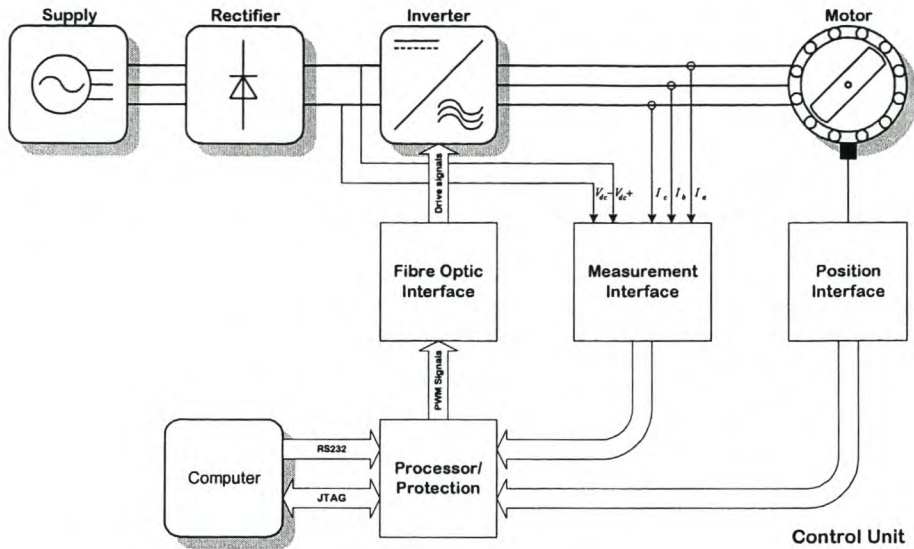


Figure 6.6 - DSP Control Unit

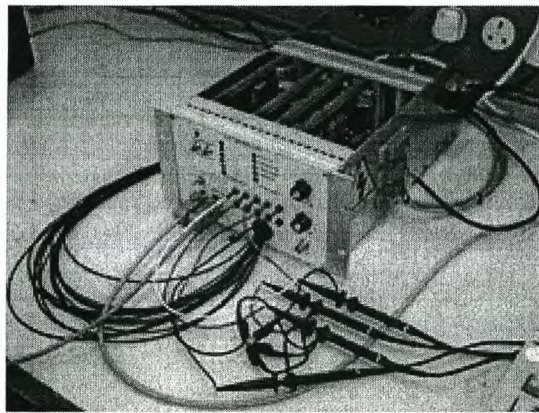


Figure 6.7 - DSP Controller

6.4.1 Processor/Protection Card

The processor block is the main card of the controller. The DSP is connected through a data bus with the other cards. This card contains the following parts:

- A 20MHz Fixed Point Digital Signal Processor
- A Programmable Logic Device (EPLD)
- Four Digital-to-Analog Outputs
- RS232 Communication port
- JTAG Communication port

These parts are discussed in more detail in the following sub-sections as shown in Fig. 6.8.

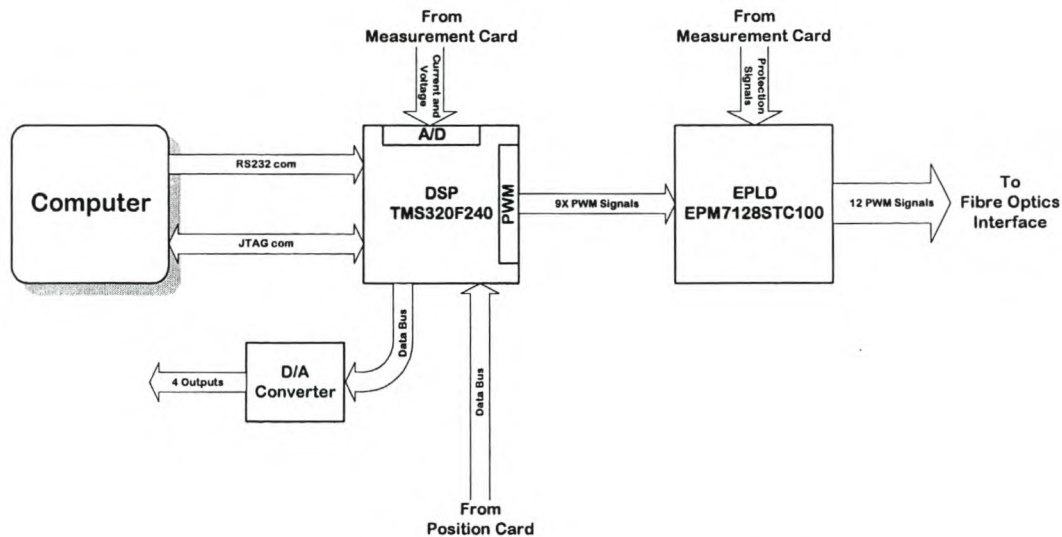


Figure 6.8 - Layout of Processor/Protection Card

a. Communication interface (JTAG and RS232)

The processor has two communication ports through which it can be programmed. The first is via the JTAG port. With this port there is direct access to the FLASH RAM as well as the registers of the DSP. However a JTAG cable and computer card is necessary for this. This port is used to initialise the processor and to load its serial port's loader program into the FLASH RAM of the DSP. This program is necessary for the communication using the second port, the serial port. This is a RS232 communications port and is connected directly to the COM1 serial port of the computer. This port is used to download the control software onto the DSP.

While using the controller, there must also be communication between the processor and the user so that the processor can give information of what is happening and at what point in the program the processor is. For this there are two types of outputs and inputs available. The first is the standard I/O pins to the processor that is accessible from the front panel of the controller. The second is four analog channels from the four DAC's connected to the DSP's data bus. With these analog channels every variable or counter inside the DSP can be monitored for debugging.

b. The Processor Circuit (DSP TMS320F240)

This circuit (see Fig. 6.8) mainly consists of the DSP processor together with its inputs and outputs as well as the communication ports, which were discussed in the previous section. The most of the inputs to the DSP are digital. There are three I/O ports available. The 16-bit data bus and address bus is also available on the back plane of the euro rack for use by the position card.

The DSP have sixteen analog-to-digital channels that are connected via the universal bus to the other interface cards of the controller. All the measurements as well as the inputs from the user are connected to these channels. The channels are divided into two groups. The first eight channels are connected to the first ADC and the remaining eight are connected to the second ADC. Thus these channels are multiplexed through two ADCs. The control of the ADCs is done with the software program of the DSP.

With regard to the outputs the DSP has twelve PWM / Compare output pins. These are divided into three groups of PWM / Compare pins. The first is the full compare group. This group consists of 3 pairs of PWM pins, one of the general purpose timers and the dead time unit. The unit generates six PWM outputs for a half bridge configured inverter. Three PWM signals are for the top IGBTs and three for the bottom IGBTs and all six have dead time as programmed in the software program. The second unit is the simple compare unit. This unit is the same as the full compare unit, but has only 3 PWM / Compare pins and does not have a dead time unit. This unit is connected to the second counter. The last three PWM pins are standalone and can be used to do anything. One of these is connected to each of the three general-purpose counters.

All of the compare units are controlled independent of the program loop. Thus the output is continuous, except in the case of a fault in which case there is a hardware interrupt line that if active will pull the PWM pins low and will in-effect switch all the IGBTs of the inverter off. The PWM signals are connected through the protection algorithm of the EPLD (see Fig. 6.8) and in the case of a fault the PWM signals can also be disabled there.

c. Protection Circuit (EPLD EPM7128STC100)

The protection circuit consists of the programmable Logic Device (EPLD) as shown in Fig. 6.8. The DSP supply the EPLD with nine PWM signals. By using a selective six signals, either six- (Bipolar switching) or twelve- (Unipolar switching) signals can be given to the inverter. Furthermore, dead time is added to all the drive-signals to prevent top and bottom IGBTs switching together and accidentally short-circuiting the DC bus voltage. This logic controller also receives error signals from the Fibre Optic-, Measurement- and Position interfaces. These error signals are as follows:

- Over-current error (On I_a , I_b and I_c)
- Over-voltage error (On V_{dc})
- Over-speed error
- Error from the inverter

On any of these errors above, the EPLD will immediately switch off all the drive signals to the inverter.

6.4.2 Measurement Card

On the electrical side there are a few parameters to be measured. This card makes provision for five current measurements, six voltage measurements, and one DC-bus voltage measurement as shown in Fig. 6.9. The measurement circuits are described in the following sub-sections.

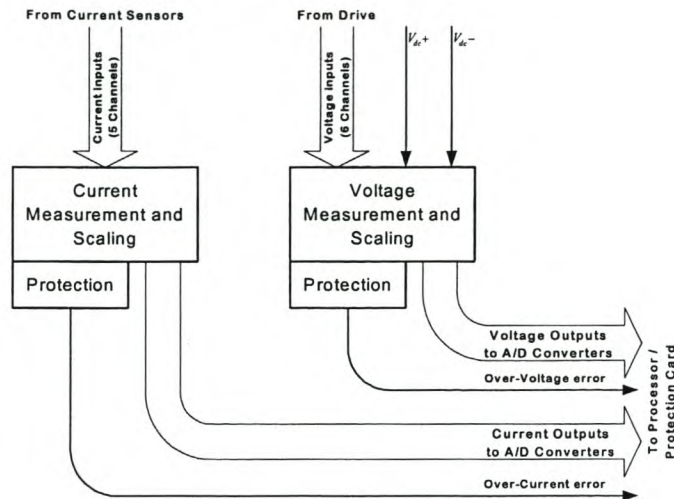


Figure 6.9 - Layout of Measurement Card

a. Current Measurement

There are current measurements with and without over current protection. The first three measurements (I_a , I_b and I_c) have fault detection. This protection looks at the measured current waveform and gives an error signal (see Fig. 6.9) to the EPLD when the current crosses the minimum or maximum safety limits that are set by adjusting the two voltage levels via potentiometers. The other two current measurements do not have the error detection and are for future applications for the DSP control unit. The circuit diagram for the current measurement is shown in Fig. 6.10.

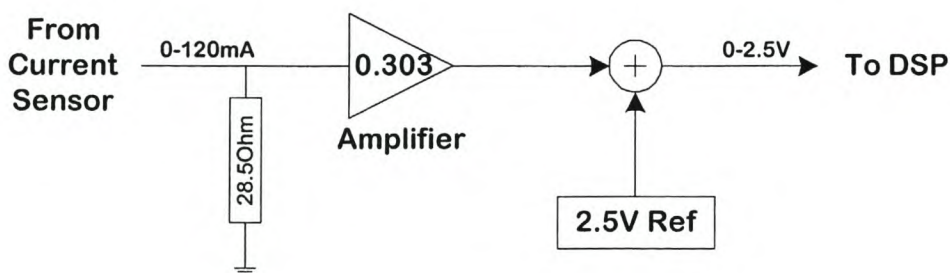


Figure 6.10 - Current measurement circuit.

The current in the three-phase motor cables are measured by means of “LEM” Hall-effect current transducers. This transducer used is rated at 300 A rms maximum, and has a conversion ratio of 1:2500. Thus, the secondary current is 120 mA rms

maximum. With a termination resistance of about 68.5Ω , the input voltage varies between 8.22V and -8.22V. The input voltage range for the ADC is 0V – 5V. Because of this, the input signal is amplified and a 2.5V reference voltage is added to comply with the voltage range requested by the ADC.

b. Voltage Measurement

With the voltage measurement there are six channels, but none of the channels have over voltage protection. Software protection can be added within the DSP program if necessary. The voltage measurements are added for future applications for the DSP control unit. The circuit for the voltage measurement is shown in Fig. 6.11.

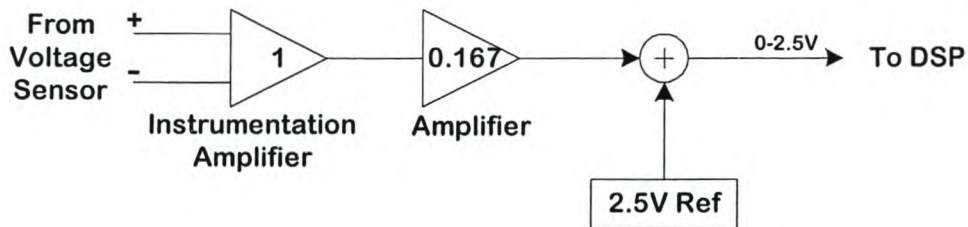


Figure 6.11 - Voltage measurement

The circuit for the measurement is the almost the same as in the case of the current measurement with the exception of the first stage. The high voltage sensor is placed directly on the measurement card. There are high voltage connectors on the side of the control unit that are capable of handling voltages up to 1000V. The voltage sensor consists of a resistive voltage divider circuit. The output voltage of the sensor can be adjusted for a certain voltage range. However, the higher the voltage range, the lower the resolution of the signal inside the DSP. This voltage from the sensor is measured with an instrumentation amplifier and then scaled for the ADC, like discussed with the current measurements and this scaling influences the resolution of the signal.

c. DC bus voltage Measurement

The last part of the measurements is the dc-bus voltage measurement. This measurement is done on the same principle as with the voltage, but it has error detection for protection. Under regeneration conditions of the drive system the DC

bus voltage will increase because the rectifier is a passive diode rectifier and is therefore not bi-directional for current flow. This will cause the DC bus voltage over the dc-bus capacitors to increase. The protection detects the level of the dc-bus voltage and if higher than the maximum limit, it gives an error signal to the EPLD.

6.4.3 Position Card

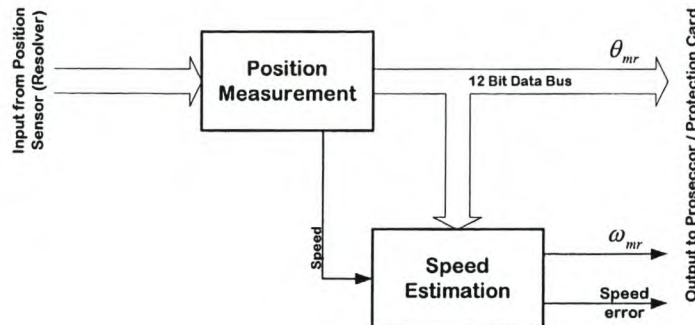


Figure 6.12 - Layout of position card

The Position card is the second measurement card of the control unit. A layout is given in Fig. 6.12. The difference with this measurement with respect to the current- and voltage measurements is that the position is digitally measured and transmitted to the DSP. The position sensor used is a brush-less resolver together with its controller circuit. The controller circuit can be selected to give different resolutions of position. It can give 10-, 12-, 14- or 16-bit resolution. For this application the 12-bit selection is used which gives an accurate position of the rotor. The 12-bit digital position is directly read into the data bus of the DSP.

For speed estimation, two hardware methods are used. The first is the use of the velocity output of the resolver controller chip itself. For the second method the least significant bit of the 12-bit position is used. This signal is put through a frequency-to-voltage converter that gives the velocity as an analog voltage signal. Each of these two methods can be selected. The speed is then scaled and connected to one of the DSP's ADC channels.

There is also a protection on the speed of the system. If the speed increases above the protection level an error signal is send to the EPLD, which then switch off the IGBTs of the inverter.

6.4.4 Fibre Optic Interface Card

The Fibre Optic Card is in fact the interface between the controller and the inverter. This interface also isolates the controller and the inverter electrically, because the controller signals that switch the different IGBTs are transmitted via optic light through optic cables from the controller to the inverter. The card converts the six or twelve drive signals from digital electric signals to optic signals. On the inverter side the inverse is done to obtain the switching states for the IGBTs.

There are also three optical error signals coming back from the inverter. They are received by the interface card and combined into a single electric error signal that is send to the protection card.

7 - Simulation and Practical Evaluation of the Dynamic Control Techniques

The design and implementation of the software and construction of the hardware of the controller are discussed in detail in the previous chapters. This chapter focuses on the results of the simulation and the practical measurements of the RSM drive system itself. Both the speed control methods dealt with in Chapter 5, using the constant- d -axis and constant-current-angle current control methods, are evaluated in this Chapter. These two methods are compared in performance and efficiency with each other.

7.1 Constant- d -axis-current speed controller

The constant- d -axis-current speed controller has two inputs that can influence the speed (see Fig. 7.1). The first is the speed reference input for which the PI controller is designed to compensate for. The second is the load torque input. The PI controller is not primarily designed for the load torque input, but will compensate for the speed variation due to a change in load torque. The designed controller is tested and simulated (see next two sections) for step responses in each of these inputs. A block diagram of the speed controller is shown in Fig. 7.1.

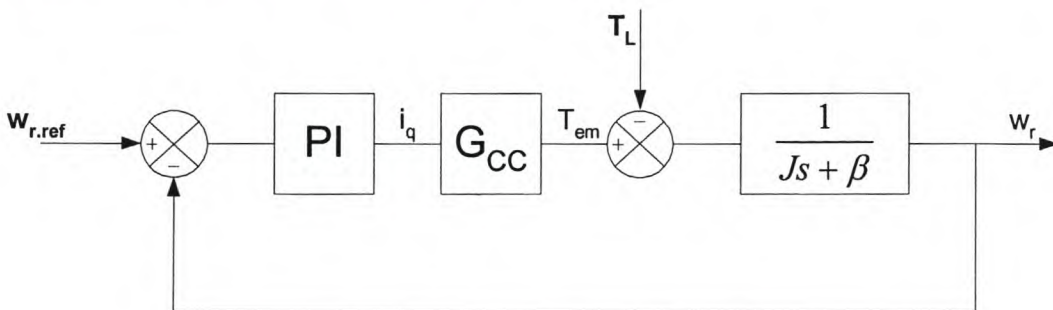


Figure 7.1 - Basic block diagram of the constant d -axis speed control loop

For the practical measurement the DSP signals are output through the digital-to-analog converters, and then filtered and averaged by a 150MHz oscilloscope to minimise to some extent the switching noise induced at the measuring channels of the oscilloscope.

7.1.1 Testing of the speed controller

In the simulation a step in the speed reference input from 0 to 1000 rpm is given to the system and the load torque is kept at zero. The response for this speed input is obtained and shown in Fig. 7.2 below. The reference speed and the actual speed are in terms of mechanical speeds.

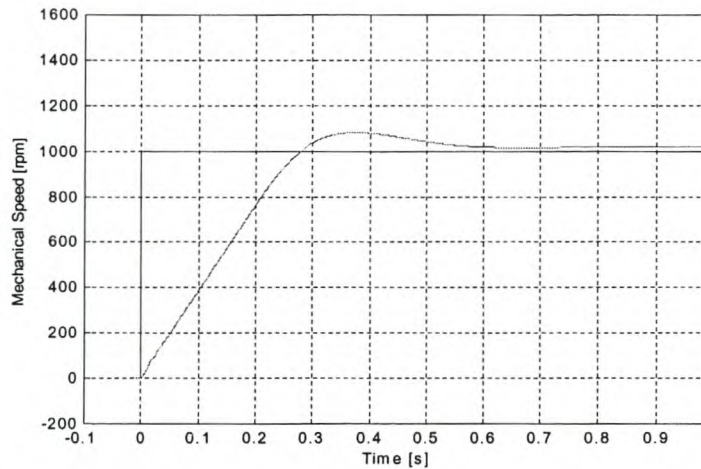


Figure 7.2 - Simulated step response in speed of the RSM drive.

From this response it can be seen that the settling time of the simulation is in the region of 500 milliseconds. This corresponds with the designed response time. The response has further a 6% overshoot, which is also in agreement with the slightly under damped designed response.

As discussed and shown in the results of the current controller of Chapter 4, there is a constant offset between the reference speed and the actual speed in the steady state (see Fig. 7.2). This is due to loss in resolution caused by the scaling and truncating of values. This is a result of the fixed-point calculation of the processor used to implement the digital controller. The scaling and the truncation are incorporated into the simulation to obtain an accurate representation of the actual drive system.

The same speed step as for the simulation is applied to the actual system, and the speed reference and the rotor speed are measured. This response is shown in Fig. 7.3.

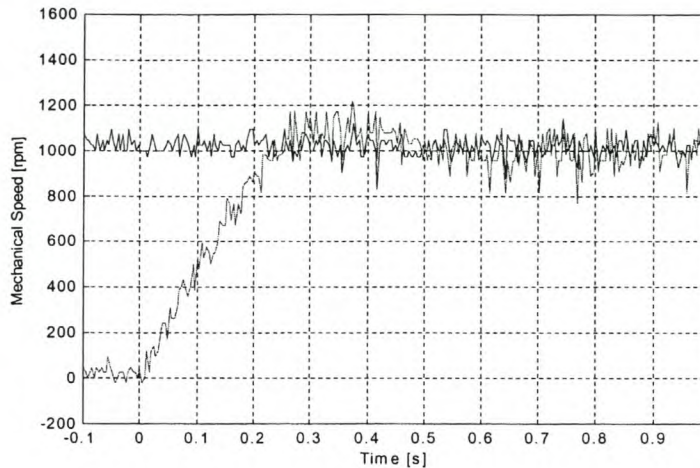


Figure 7.3 - Measured step response in speed of the RSM drive.

The settling time of the speed is about 500 milliseconds and has an overshoot of about 6%. This corresponds with the response of the simulation and the design response. This response is much faster than the natural response time of the open loop system, which is about 10 times slower.

The d - and q -axis currents were also measured during the step response of the speed. The current measurements are shown below in Fig. 7.4.

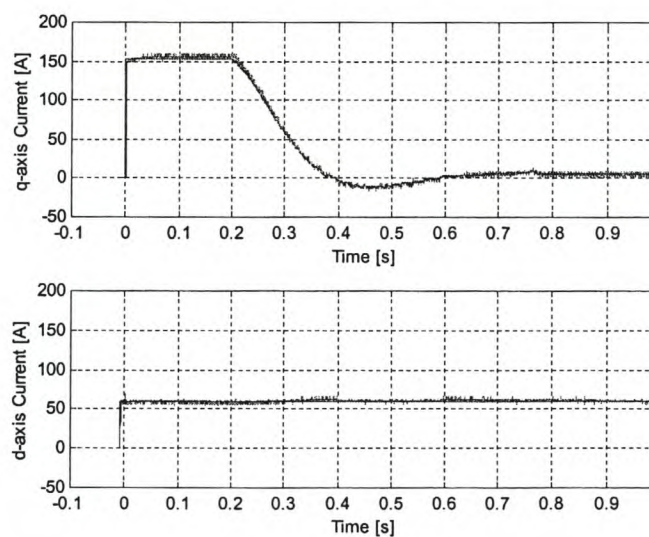


Figure 7.4 - Simulated response of d - and q -axis currents for constant- d -axis-current speed control

The d -axis current is kept constant and it can be seen from Fig. 7.4 that the d -axis current is injected into the machine a few milliseconds earlier than the step command of the speed. This is to ensure that the machine is at full flux before the speed

command is given. With the step speed command, q -axis current is injected, but also limited to its rated value to limit the torque of the machine to its rated torque. While in the limit region, the machine will constantly accelerate, as seen in Figs. 7.2 and 7.3, and near the desired speed the current comes out of the limited region and the PI controller begins to control the speed (see Fig. 7.4). The integration part of the PI controller is halted during limiting of the current. This can be seen from the simulated results of the currents. The measured results from the actual system are also obtained and are shown in Fig. 7.5.

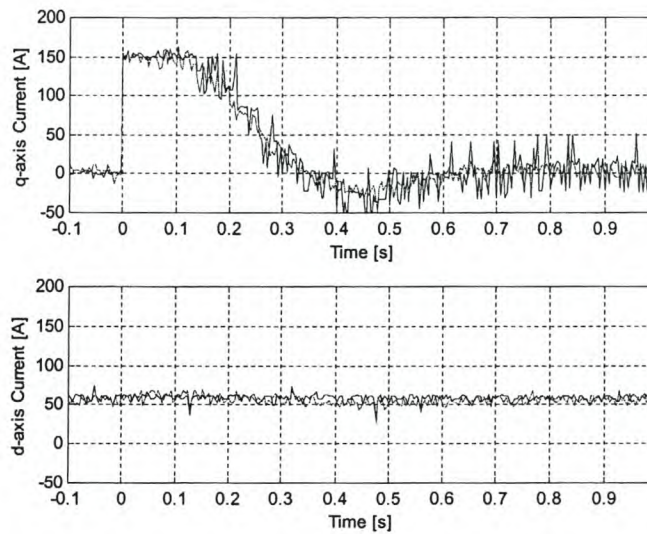


Figure 7.5 - Measured response of the d - and q -axis currents for constant- d -axis-current speed control

The measured response corresponds very well with the simulated response. The time in the limited region is a bit less than the simulation but it can be due to small varying factors that are not taken into account in the simulation. There is almost no influence of the cross-magnetization and speed voltage on the d - and q -axis currents. The step in speed can also be done from a 1000rpm to 0rpm. This test was not done due to the fact the power electronic system can not regenerated power back to the supply and is limited in dissipating the power into the dc-link resistive load.

7.1.2 Load testing of the system

The controller performs well with the step response in speed. In the load testing, the machine is held at a constant speed and a step is given in the load torque. This will

show the performance of the speed controller with a load disturbance in the control loop. The machine is kept at its rated speed of 1500 rpm and is given a load step of 200 Nm developed by the dynamometer-load. The full-load torque of the machine is about 300 Nm.

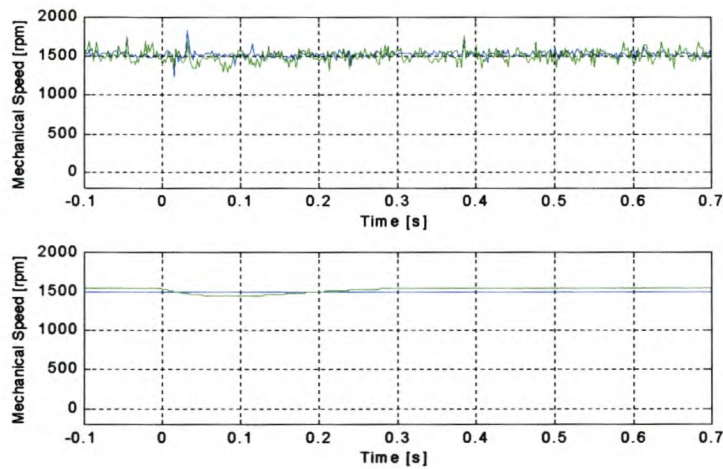


Figure 7.6 – Constant-d-axis-current speed control response with a 0 – 200 Nm load step.

(top : measured; bottom : simulated)

Figure 7.6 above shows the reaction of the speed controller to a change in load. The simulated reaction is the bottom waveform and the measured response the top one. The speed decreases with the load step but recovers to the reference speed within 250 milliseconds. The drop in speed is about 6.3% (91rpm) and the speed controller is effective in preventing a large variation from the reference speed. The q -axis current reference of the PI controller is also monitored with the step in load. The measured and simulated currents are given in Fig 7.7. The measured current reference has a smaller overshoot than the simulated current reference.

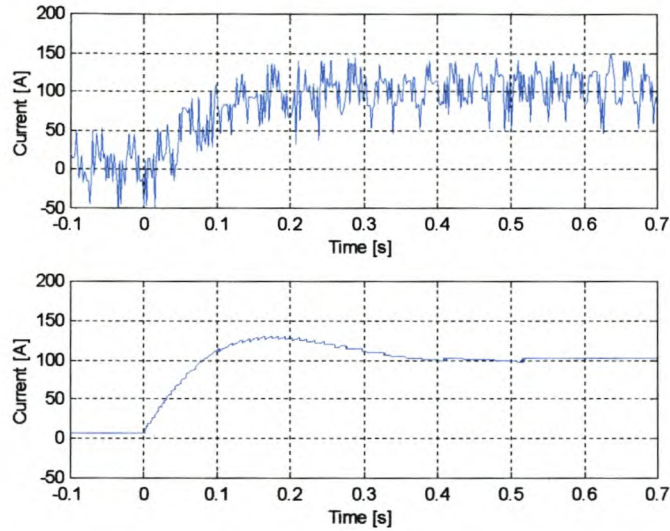


Figure 7.7 - Current reference (q -axis) during step in load torque

(top : measured; bottom : simulated)

Together with the positive step in the load torque, negative step in the load torque is also given. The load applied to the machine in the previous test is now removed and the response of the speed of the machine is measured and shown in Fig. 7.8.

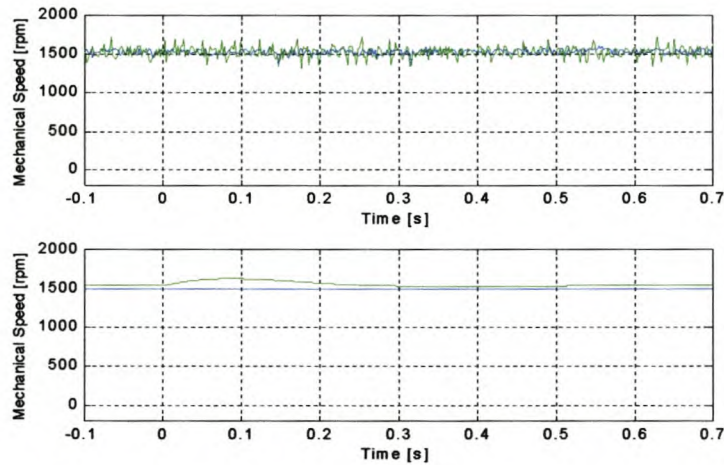


Figure 7.8 – Constant- d -axis-current speed control response with a 200 – 0 Nm load torque step

(top : measured; bottom : simulated)

With the removing of the load, the machine tends to accelerate and the controller compensates for this by extracting current from the machine to keep it at same speed as before. Looking at the measured and simulated results in Fig. 7.8, it can be seen that the controller reacts within 250 milliseconds with a 6.01% (93rpm) increase in

speed. Thus the same response with the negative load step is obtained than that obtained with the positive load step.

The difference between the measured and the simulated speed responses, with positive and negative load torque steps can be due to the fact that the load steps of the dyno on the machine is slower than that in the simulation. The simulated steps in load torque are instantaneous.

7.2 Constant-current-angle speed controller

This speed controller has the same transfer function as that of the constant- d -axis-current controller, with the exception of the transfer function of the current controller G_{cc} in the block diagram of Fig. 7.9. It has a speed reference input and load torque input, but the output of the PI controller controls the magnitude of the current vector and not the magnitude of the q -axis current. This controller is tested in the same way as the constant- d -axis-current speed controller.

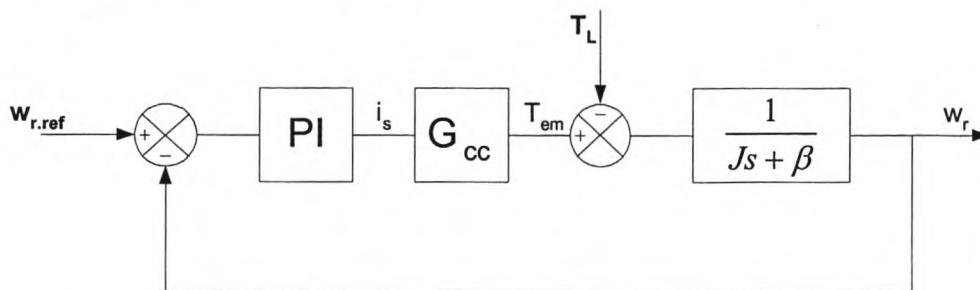


Figure 7.9 - Basic block diagram of the constant-current-angle speed control

7.2.1 Testing of the speed controller

The load of the machine is kept zero and a step is applied to the speed reference in the simulation. The response of the control loop to the step in speed is observed and is plotted against the reference speed in Fig. 7.10.

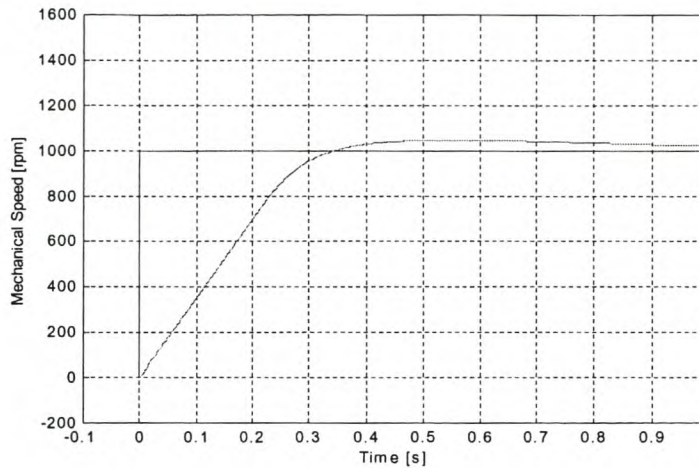


Figure 7.10 - Simulated step response of the speed (constant-angle speed controller)

The response is almost critically damped with a small overshoot. The settling time of the response is about 500 milliseconds, which corresponds well with the designed settling time of 500 milliseconds. The actual speed of the machine has a constant offset with respect to the reference speed (see Fig. 7.10). This is because of scaling and truncating of the DSP values, like in the case with the constant- d -axis-current speed control. The offset, however, is bigger than the offset of the constant- d -axis-current speed controller, because of the scaling and truncating in the conversion from the magnitude and angle of the current vector to the d - and q -axis reference currents. This offset is not dependent on the speed and, thus, will not change with an increase or decrease in speed.

The step response of the actual drive system is measured and is given in Fig 7.11. This response has a settling time that is almost exactly the same as that of the simulated response. The simulated results and the practical results are very close agreement.

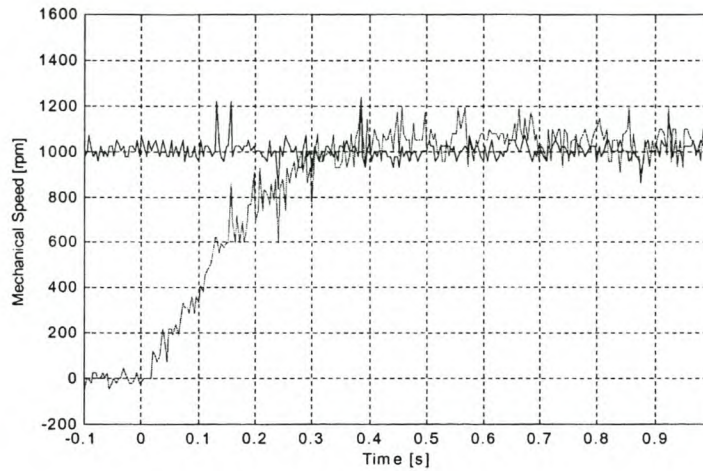


Figure 7.11 - Measured step response of the speed (constant-angle speed controller).

The simulated d - and q -axis currents, with the step response in speed, are given in Fig. 7.12.

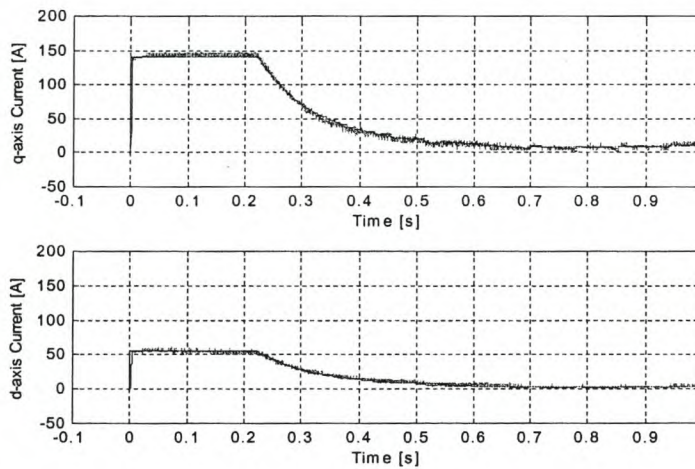


Figure 7.12 - Simulated response of d - and q -axis currents for constant-current-angle speed control

In the control both the d - and q -axis currents are injected into the machine at the same time. This means that there is no flux in the machine before the step command. The magnitude of the current vector is limited to limit the torque of the machine to rated torque. The integration part of the PI speed controller is halted when the current is limited. The measured results are shown in Fig. 7.13.

The response of the practical system corresponds well with the results obtained from the simulation.

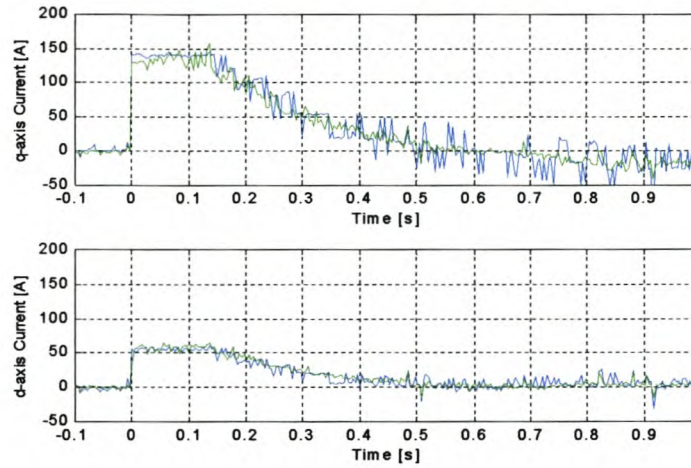


Figure 7.13 - Measured response of d- and q-axis currents for constant current angle speed control

7.2.2 Load testing of the system

For the load testing of the RSM drive the machine is controlled to run at a constant speed. A load step is applied by means of the eddy-current dynamometer. The load step applied is a 0 to 200 Nm load step while the machine is running at 1500rpm. This is the same load applied as to the constant-*d*-axis-current speed controlled system in section 7.1.2. The reference and actual speeds are simulated and practically tested. The results are shown in the Fig. 7.14.

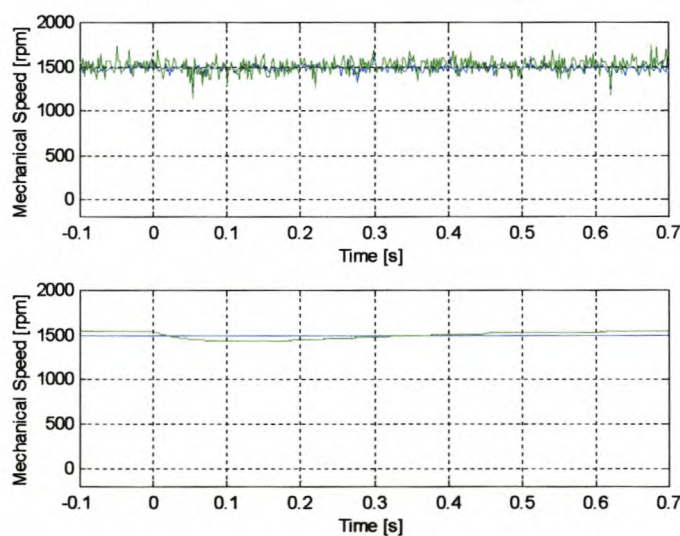


Figure 7.14 – Constant-current-angle speed control with a 0 – 200 Nm load step

(top : measured; bottom : simulated)

The response time of the controller is about 450 milliseconds and is sufficient for such a load step. There is also a small drop in the speed shown in the simulation. Due to the fact that the load step response of the dynamometer is not instantaneous as in the simulation, the drop in the speed is less in the practical measurements and is not clearly visible.

A negative step in the load is also applied and the response evaluated. The machine is operated at a constant speed of 1500 rpm with a constant load of 200 Nm. The load is then removed and the reference and actual speed are observed. The results are shown in Fig. 7.15.

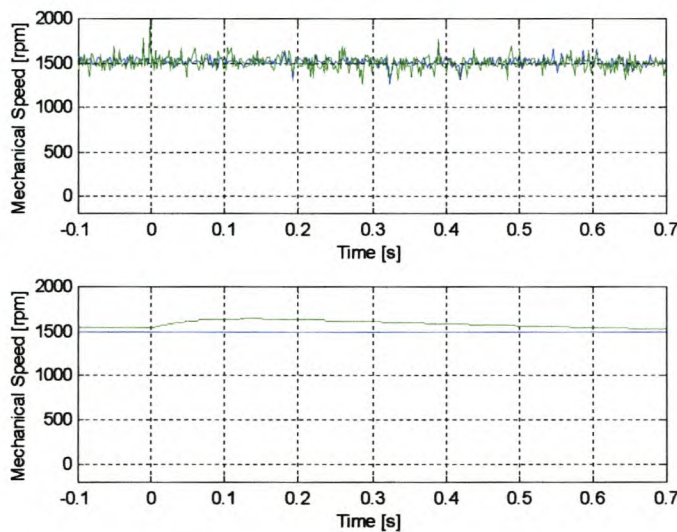


Figure 7.15 – Constant-current-angle speed control with a 200 – 0 Nm load step.

The simulation results show that the speed of the machine increase with the removal of the load. The settling time of the system with the load change is about 500 milliseconds. This corresponds well with the response time of the controller that is also about 500 milliseconds. The positive and the negative step response are also the same. Again the measured results show almost no change because of the load step not being instantaneous.

7.3 Torque ripple compensation

There is always torque ripple present on the developed torque of the machine. This ripple can be minimized in the design of the RSM by skewing the rotor by one or more slot pitch. A slot pitch is the angle between two stator slots. The RSM used in this project has an unskewed rotor and its torque ripple is calculated by using the finite element analysis. This is shown in Fig. 7.16. At full-load torque the machine has a torque ripple of about 200 Nm peak-to-peak. The torque ripple, as a function of the rotor position and the flux, is included into the model of the RSM. There are a few methods, as given in [20] and [21], that can be implemented in the controller to compensate for this torque ripple disturbance.

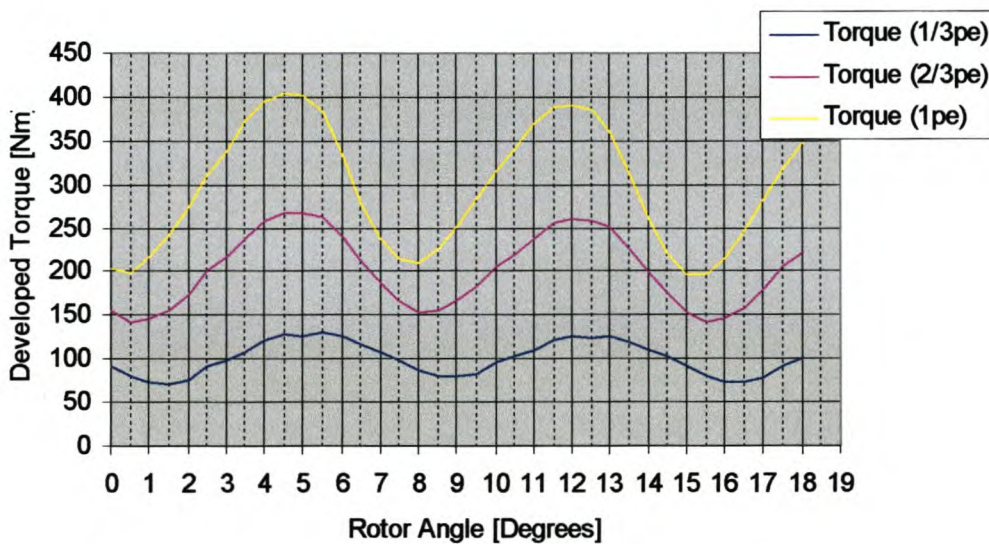


Figure 7.16 - Torque ripple of the RSM

As mentioned in Chapter 5, the speed controllers are designed for a step response in speed. The previous sections show the influence a step in the load torque on the speed controller. To compensate for the load fluctuations, the compensation method of Matsui [20], given in Fig. 7.17, is evaluated by simulation on the constant- d -axis-current and the constant-current-angle speed controllers.

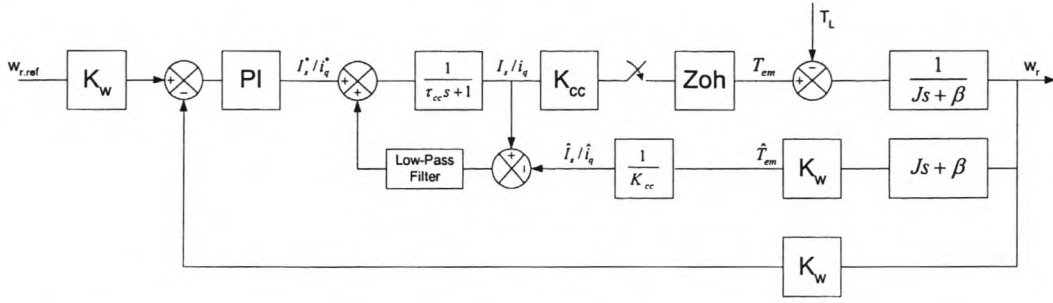


Figure 7.17 - Block diagram of the speed controller with torque ripple compensation

The part in grey of the block diagram in Fig. 7.17 is the torque ripple compensator.

The inverse of the mechanical transfer function of the machine is given as

$$G(s) = Js + \beta, \quad (7.1)$$

which is transformed to the z-plane as

$$G(z) = \left(\frac{J}{T} + \beta\right) - \frac{J}{T} z^{-1}. \quad (7.2)$$

where J and β in the inertia and the friction respectively and T the sampling time.

Using eqn (7.2) the estimated torque \hat{T} is obtained from the speed. From this estimated torque the estimated current \hat{I}_s (constant-current-angle control) or \hat{i}_q (constant- d -axis-current control) in the machine is determined by dividing with the current-to-torque constant, K_{cc} . The difference between the actual current and estimated current is then filtered and added to the current reference.

The constant- d -axis-current speed controller is evaluated with and without the compensator. A step in speed from 0 to 300 rpm is given and after 1.5 seconds a 200Nm step in load torque is given to the system. The speed response is shown in Fig. 7.18.

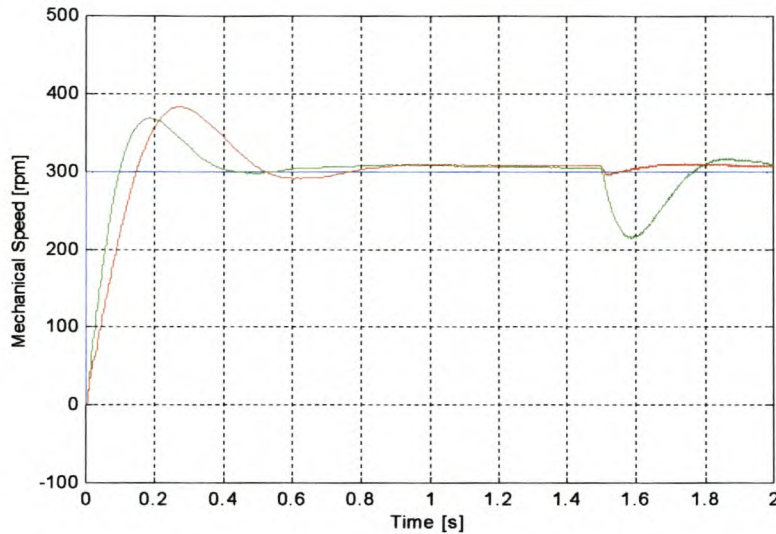


Figure 7.18 - Simulated speed with (red) and without (green) ripple compensation

The response of the speed controller (see Fig. 7.18) is slower and has a larger overshoot with the ripple compensator, but the influence of the load torque on the speed controller is much lower than without the compensation. The increase in response time of the speed control loop can be due to the low-pass filter used with the compensation. The type of low-pass filter must be investigated.

The q -axis current, with the same step in the speed and torque as shown in Fig. 7.18, is shown in Fig. 7.19. The current response with ripple compensation (bottom) is much faster than the current without the compensation.

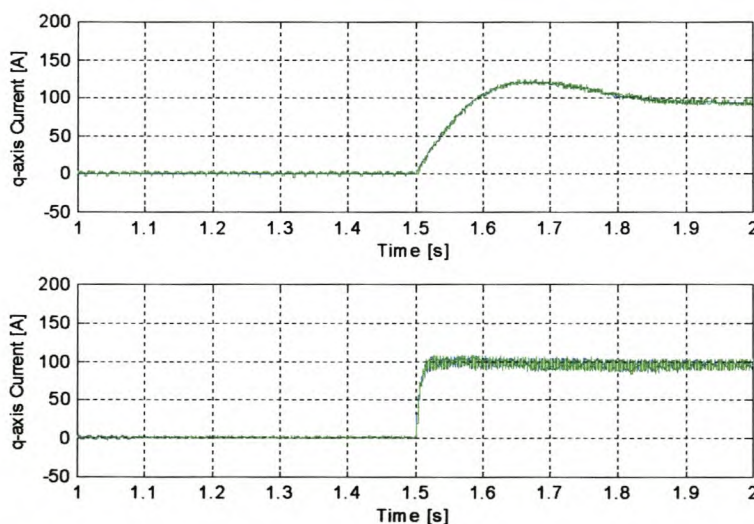


Figure 7.19 - q -axis current with (bottom) and without (top) ripple compensation

The ripple compensation is evaluated with constant-current-angle speed control. The same step commands in speed and torque is given as with the constant-d-axis-current speed control and the response of the speed is shown in Fig. 7.20.

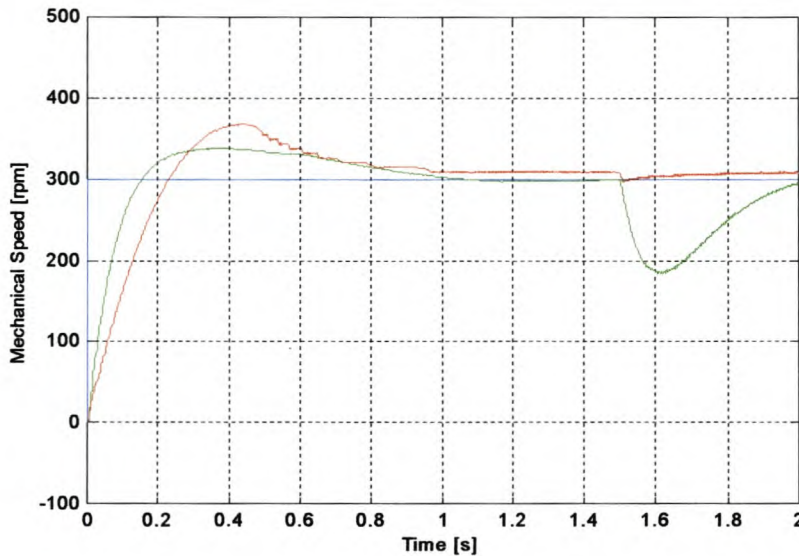


Figure 7.20 - Simulated speed with (red) and without (green) ripple compensation

The response, with ripple compensation, with the change in the speed reference is slower and has a larger overshoot than without the compensation. The load step has less influence on the speed controller with the compensation than the speed controller without compensation. The current response is given in Fig. 7.21. The compensated current response is much faster than the response without the compensation.

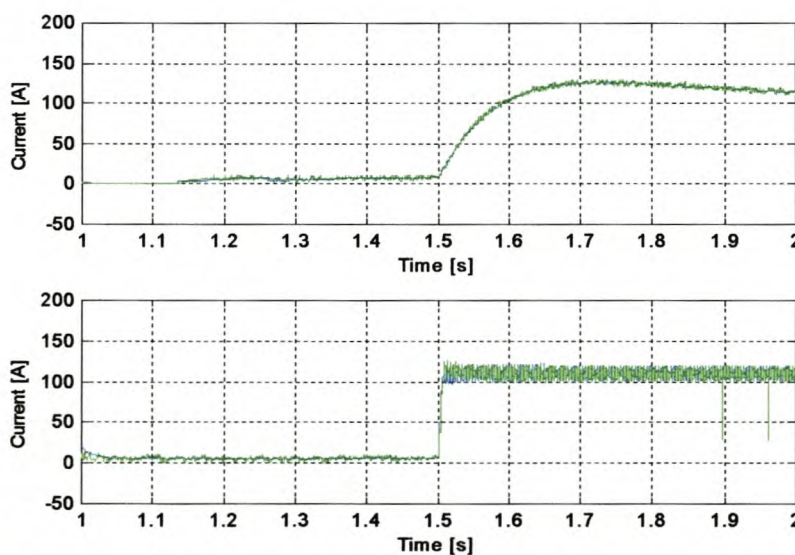


Figure 7.21 – I_s current with (bottom) and without (top) ripple compensation

7.4 Comparison of the control methods

The performance of the constant- d -axis-current controller and the constant-current-angle speed controller are both evaluated in the previous sections. Keeping the dynamic performance as well as the energy-efficiency in mind, these two control methods are compared in this section on the following grounds:

- Current response
- Speed response
- Power efficiency

The dq -axis current controller of both the two control methods is the same as designed in Chapter 4. For the constant- d -axis-current control, the d -axis current is constant and, thus, the step response time depends only on the response time of the q -axis current controller. This response time is in the region of 2 ms as shown in Fig. 5.2. In the case of the constant-current-angle control, both the d - and q -axis current controller response times are effecting the response time of the controller. Due to the fact that the d -axis current response is slower than the q -axis current response, the overall current response is equal to the d -axis current response. This response is about 4 ms as shown in Fig. 5.7. Hence, the constant- d -axis-current control is 2 ms faster (or else twice as fast) than the constant-current-angle control. This difference is very small and both methods have good dynamics.

Looking at the speed response of the constant- d -axis-current speed control as evaluated in section 7.1, the response time for a step in speed from 0 to 1000 rpm is about 500 milliseconds (see Figs. 7.2 and 7.3). The same step is applied to the constant-current-angle speed control and a response of 480 milliseconds was obtained. Thus the speed responses of both control methods are almost identical. This means that the overall dynamics of the two control methods are the same. This is expected, as the speed dynamics is much-and-much slower than the current control

dynamics. The double as slow constant-angle current control, thus, will not have any effect on the speed dynamics.

From a power efficient point of view the current of the machine is calculated for both of the control methods under no-load and under two-thirds of the load and is given in Table 7.1.

	Is (A)	
	no-load	2/3 load
Constant-d-axis-current control	60.2	118.3
Constant-current-angle control	12.6	116.9

Table 7.1 - Current in the RSM

In the case of the constant-d-axis-current control, full d -axis flux is present in the machine under no-load and full-load conditions. Thus, under no-load conditions there are copper and iron losses in the machine. On the other hand with the constant-current-angle control the stator current and the flux in the machine at no-load is practically zero. This means that the losses in the machine are practically zero. This shows that the constant-current-angle control is overall much more energy-efficient than the constant- d -axis-current control.

8 - Conclusions and Recommendations

The conclusions reached in this Chapter are based on the implementation and evaluation of the constant-current-angle control in comparison with the conventional constant- d -axis-current control in terms of dynamic performance and energy efficiency. Recommendations are given to further investigate the constant-current-angle control method as applied to the RSM drive.

8.1 Conclusions

From the results the conclusions reached are summarised as follows:

- The influence of cross-magnetization and speed voltages on the current controllers can be minimised by matter of decoupling. It is shown that with decoupling accurate dynamic control of the current according to design is obtained.
- There are various complex vector control methods proposed for better dynamic performance and efficiency of RSM drives. The constant-current-angle control however is simple, has good dynamic performance and is also energy efficient. The dynamic performance of the constant-current-angle control method compares well with that of the constant- d -axis-current control method. As an application it is shown how the constant-current-angle control method can be used for torque ripple compensation in a drive system.
- It is shown that the inexpensive fixed-point TMS320F240 DSP-based digital controller is well suited for implementation of a full constant-current-angle speed control scheme. However it is shown that care must be taken with regards to scaling and truncation in the DSP control program.

8.2 **Recommendations**

Only the following two recommendations are made with respect to the constant-current-angle controlled RSM drive:

- I. It is recommended that the use of position sensorless control using constant current angle control be investigated and implemented on a RSM drive.
- II. Further improvement of the DSP controller must be investigated with regard to:
 - a. speed measurement,
 - b. the use of space vector modulation in stead of PWM control, and
 - c. the use of a floating-point processor to avoid scaling and truncation effects.

Bibliography

- [1] **Germishuizen** J.J., “*Comparative study of Reluctance Synchronous and Induction Machine Drives for rail Traction*”, M.Sc. Eng. dissertation, University of Stellenbosch, March 2000.
- [2] **Matsuo** T., Lipo T.A., “*A New Control Strategy for Optimum-Efficiency Operation of a Synchronous Reluctance Motor*”, IEEE Transactions on Industry Applications, Vol.33, No. 5, Sept/Oct 1997, pp.1146-1153.
- [3] **Matsuo** T., El-Antably A., Lipo T.A., “*Field Oriented Control of Synchronous Reluctance Machine*”, IEEE Power Electronics Specialists Conference Record, June 1993, pp.425-431.
- [4] **Vagati** A., Pastorelli M., Farceschini G., “*High Performance Control of Synchronous Reluctance Motor*”, IEEE IAS Conference Record, Oct 1996, pp.295-303.
- [5] **Fratta** A., Vagati A., “*A Reluctance Motor Drive for High Dynamic Performance Applications*”, IEEE Transactions on Industry Applications, Vol. 28, No.4, July/Aug 1992, pp. 873-879.
- [6] **Kamper** M.J., Mackay A.T., “*Optimum control of the reluctance synchronous machine with a cageless flux-barrier rotor*”, Transactions of the South African Institute of Electrical Engineers, Vol. 86, No. 2, June 1995, pp. 49-56.
- [7] **Kamper** M.J., Trubenbach R.A., “*Vector control and performance of a reluctance synchronous machine with flux barrier rotor*”, ICEM September 1992, pp 547-551.

- [8] **Fernández-Bermel** F., García-Cerrada A., Faure R., “*Model-Based Loss Minimization for DC and AC Vector Controlled Motors including Core Saturation*”, IEEE Transactions on Industry Applications, Vol. 36, No.3, May/June 2000, pp.755-763.
- [9] **Cupertino** F., Lattanzi A., Salvatore L., “A New Fuzzy Logic-Based Controller Design for DC and AC Impressed-Voltage Drives”, IEEE Transactions on Power Electronics, Vol.15, No.6, Nov 2000, pp.974-982.
- [10] **Lagerquist** R., Betz R.B., Miller T.J.E., “*DSP96002 Based High Performance Digital Vector Controller for Synchronous Reluctance Motors*”, ICEM Sept 1992, Vol.3, pp.903-907.
- [11] **Krause** Paul C., “*Analysis of Electric Machinery*”, Mc Graw-Hill Book Company, pp.133-148.
- [12] **Conti** G., Parasiliti F., Villani M., “*Torque Ripple Analysis is Synchronous Reluctance Motors*”, Electromotion 3, 1996, p.188-193.
- [13] **Kamper** M.J., “Design Optimisation of Cageless Flux Barrier Rotor Reluctance Synchronous Machine”, P.hD. Dissertation, University of Stellenbosch, December 1996.
- [14] **Kamper** M.J., Van der Merwe F.S., Williamson S., “*Direct finite element design optimization of the cageless reluctance synchronous machine*”, IEEE Transactions on Energy Conversion Vol. 11, No. 3, September 1996, pp. 547-555.

- [15] **Kamper** M.J., Volschenk A.F., “*Effect of rotor dimensions and cross magnetization on L_d and L_q inductances of reluctance synchronous machine with cageless flux barrier rotor*”, IEE Proc.-Electr. Power Appl., Vol. 141, No. 4, July 1994, pp.213-220.

- [16] **Fratta** A., Petrache C., Franceschini G., Troglia G.P., “*A simple Current Regulator for flux-weakened operation of High Performance Synchronous Reluctance Drives*”, IEEE IAS Conference Record, Oct 1994, pp.649-656.

- [17] **Wiley** B., Hudson R., Spée R., “*Digital Enhancement of Analog Current Regulators for Three Phase Motor Drives*”, IEEE PEC, 1995.

- [18] **Mouton** H.d.T., Enslin J.H.R., “*An Optimal On-Line-Tuning Current Regulator for High-Power IGBT Converters*”, IEEE Transactions on Industry Applications, Vol. 35, No. 5, Sept/Oct 1999, pp.1132-1140.

- [19] **Brod** D.M., Novotny D.W., “*Current Control of VSI-PWM Inverters*”, IEEE Transactions on industry Applications, Vol. IA-21, No. 4, May/June 1985, pp.562-570.

- [20] **Matsui** N., Makino T., Satoh H., “*Autocompensation of Torque Ripple of Direct Drive Motor by Torque Observer*”, IEEE Transactions on Industry Applications, Vol. 29, No. 1, Jan/Feb 1993, pp.187-194.

- [21] **Kim** K.H., Youn M.J., “*A simple and Robust Digital Current Control Technique of a PM Synchronous Motor using Time Delay Control Approach*”, IEEE Transactions on Power Electronics, Vol. 16, No. 1, January 2001, pp.72-82.

- [22] **Jackson S.K.**, “*An Investigation of Position Sensorless Control using a Medium Power Reluctance Synchronous Machine*”, M.Sc. Eng. dissertation, University of Stellenbosch, March 1999.
- [23] **Smuts J.L.**, “*Critical evaluation of a position sensorless control technique for the reluctance synchronous machine drive*”, M.Sc. Eng. Dissertation, University of Stellenbosch, March 2001.
- [24] **Vagati A.** and Lipo T.A., “*Synchronous Reluctance Motors and Drives: A New Alternative,*” IEEE Industry Applications Society Tutorial Course, 94TH8018, 1994.
- [25] **Vagati A.**, “*The Synchronous reluctance solution: a new alternative in A.C. drives*”, Proc. IEEE IECON, Vol.1, 1994, pp.1-13.
- [26] **Fratta A.**, Vagati A., “*Synchronous reluctance vs induction motor: 'a comparison*”, Proceedings Intelligent Motion, April 1992, pp.179-186.
- [27] **Boldea I.**, “*The Reluctance Synchronous Machine and Drives*”, Oxford University Press Inc., 1996, ISBN 0-19-859391-0, pp.26-38.
- [28] **Undeland T.M.**, Mohan N., Robbins W.P., “*Power Electronics – Converters, Applications, and Design*”, Second Edition, John Wiley & Sons Inc., ISBN 0-471-58408-8.
- [29] **Vas P.**, “*Vector Control of AC Machines*”, Oxford University Press Inc., 1990, ISBN 0-19-859370-8, p.117.
- [30] **Fitzgerald A.E.**, Kingsly C., Umans S.D., “*Electric Machinery*”, McGraw-Hill Book Company, 1992, ISBN 0-07-707708-3.

- [31] **Ogata K.**, “*Modern Control Engineering*”, Prentice-Hall Inc., 1997, ISBN 0-13-261389-1.

- [32] **Phillips C.L.**, Nagle H.T., “*Digital Control System Analysis and Design*”, Prentice-Hall Inc., Third Edition 1995, ISBN 0-13-317729-7.

- [33] “*TMS320 DSP Development Support*”, Digital Signal Processing Solutions, Texas Instruments Incorporated, 1997.

- [34] “*JTAG/MPSD Emulation*”, Technical Reference, Texas Instruments Incorporated, 1994.

- [35] “*TMS320C24x DSP Controllers, Peripheral Library and Specific Devices*”, Reference Set - Volume 1, Literature Number SPRU160B, Texas Instruments Incorporated, Sep 1997.

- [36] “*TMS320C24x DSP Controllers, Peripheral Library and Specific Devices*”, Reference Set - Volume 2, Digital Signal Processing Solutions, Texas Instruments Incorporated, 1997.

- [37] “*TMS320C240, TMS320F240 DSP Controllers*” Datasheets, Literature Number SPRS042A, Texas Instruments Incorporated, Dec 1997.

- [38] “*TMS320F20x/F24x DSP Embedded Flash Memory Technical Reference*”, Literature Number SPRU282, Texas Instruments Incorporated, Sept 1998.

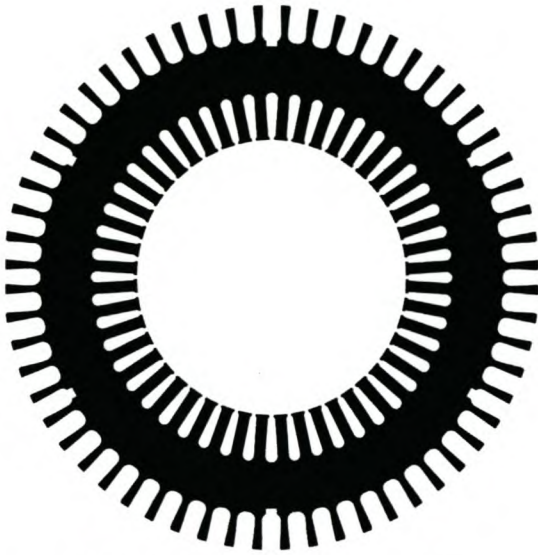
- [39] “*TMS320C2xx User’s Guide*”, Literature Number SPRU127B, Texas Instruments Incorporated, Jan 1997.

- [40] “TMS320C2x/C2xx/C5x Optimizing C Compiler User’s Guide”, Texas Instruments Incorporated, 1995.

- [41] SIMuWIN Software, Department of Electrical and Electronic Engineering, University of Stellenbosch, Version 1.0, July 1993.

- [42] MATLAB Software, MathWorks, Inc., Version 5.3, 1997.

Appendix A – Photo Album



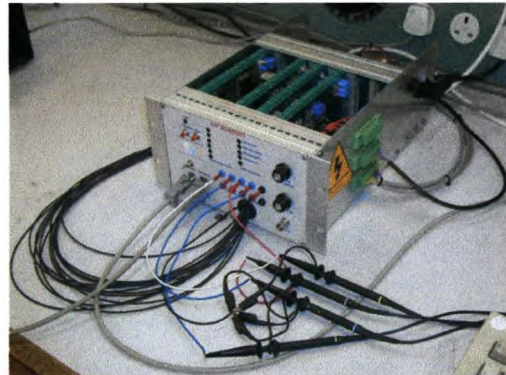
(a) Stator lamination of 42kW RSM



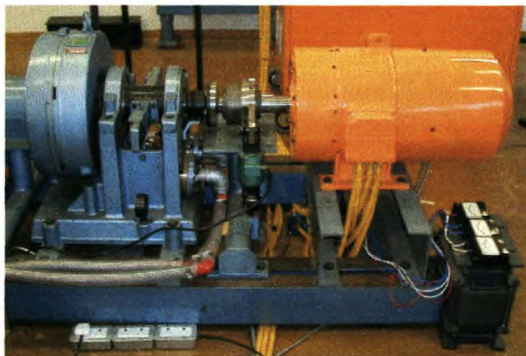
(b) Rotor lamination of 42kW RSM



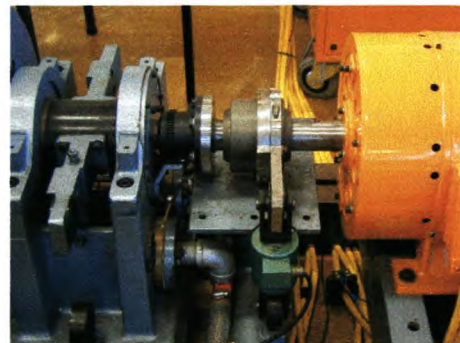
(c) 200kVA Full-bridge Inverter with Rectifier



(d) DSP Controller unit



(e) Dyno (blue) and RSM (orange) system



(f) Blocked rotor for testing

Appendix B – DSP Program for Constant-*d*-axis- Current Speed Control


```

/*****
/*      SpeedControlConstAngle.c(14 January 2002)      */
/*      */
/*      (Control program for PI speed control of the RSM) */
/* The switching is unipolar with a switching frequency of 3.33kHz. */
*****/

#include "F240.h"

volatile unsigned int *EVIMRA      = (volatile unsigned int *) 0x742C;
volatile unsigned int *ACTR        = (volatile unsigned int *) 0x7413;
volatile unsigned int *SACTR       = (volatile unsigned int *) 0x7414;
volatile unsigned int *DBTCON      = (volatile unsigned int *) 0x7415;
volatile unsigned int *COMPR1      = (volatile unsigned int *) 0x7417;
volatile unsigned int *COMPR2      = (volatile unsigned int *) 0x7418;
volatile unsigned int *COMPR3      = (volatile unsigned int *) 0x7419;
volatile unsigned int *SCOMPR1     = (volatile unsigned int *) 0x741A;
volatile unsigned int *SCOMPR2     = (volatile unsigned int *) 0x741B;
volatile unsigned int *SCOMPR3     = (volatile unsigned int *) 0x741C;
volatile unsigned int *COMCON      = (volatile unsigned int *) 0x7411;
volatile unsigned int *GPTCON      = (volatile unsigned int *) 0x7400;
volatile unsigned int *T1CON       = (volatile unsigned int *) 0x7404;
volatile unsigned int *DataIn      = (volatile unsigned int *) 0x8000;
volatile unsigned int *DataOut     = (volatile unsigned int *) 0x8001;

int  IaBack, IbBack, IcBack, IdBack, IqBack;
int  IsRef, IdRef, IqRef, Wait;
int  Va, Vb, Vc, Vd, Vq, limit, IsRefZ;
int  DAC1, DAC2, DAC3, DAC4, MdInd, MqInd, MrotorAngleZ, SpeedBack;
int  MrotorAngle, ErotorAngle, CurrentAngle, CurrentAngle2;
int  angleA1, angleB1, angleB2, angleC1, angleC2, Speed, GSBUS, SpeedBack;
int  QOut, DOut, Time, TimeTemp, PITemp, Inductance, SpeedTemp;
int  dlq_dt, dld_dt, IqBackZ, IdBackZ, MdDec, MqDec;
int  PISpeed, PISpeedIn, PISpeedInZ, SpeedRef, ESPEEDBACK, SpeedVolD, SpeedVolQ;
float PISpeedOut, PISpeedOutZ;
float  PlqOutZ, PlqOut, PlqInZ, PlqIn, PldOutZ, PldOut, PldInZ, PldIn;
extern int  sine(), md(), mq(), fluxd(), fluxq();

void main(void)
{
    /* Initialize the interrupt routines */

    *EVIMRA |= 0x0001;    /* Enable the PDPINT interrupt routine */

    /* Initialize the PWM and dead time routines */

    *ACTR  = 0x0999;      /* Make PWM1, PWM3 and PWM5 active low */
                        /* Make PWM2, PWM4 and PWM6 active high */
    *SACTR = 0x0015;      /* Make PWM7, PWM8 and PWM9 active low */
    *DBTCON = 0x0000;      /* Dead time = 0us, all 3 outputs active */
    *COMCON = 0x0307;      /* Initialize Compare PWM mode */
    *COMCON = 0x8307;      /* Initialize Compare PWM mode */

    /* Initialize the counters */

    *T1PR = 3000;          /* Period = 2*(50ns x 3000) = 300us */
    *T1CNT = 0;            /* Starting value for counter = 0 */
    *T1CON = 0x0A802;      /* Initialize Counter #1 */
    *T1CON = 0x0A842;      /* Initialize Counter #1 */

```



```

/* Initialize the ADs */

*OCRA |= 0x070F;          /* Make ADC0, ADC1, ADC8 and ADC9 active */
ADCTRL2 -> ADCPSCALE = 0; /* Pre-scale value = 0 */
ADCTRL2 -> ADCEVSOC = 0;
ADCTRL2 -> ADCEXTSOC = 0;
ADCTRL1 -> suspend_free = 0;
ADCTRL1 -> suspend_soft = 1;
ADCTRL1 -> ADCIMSTART = 0;
ADCTRL1 -> ADCINTEN = 0;    /* Disable interrupt */
ADCTRL1 -> ADCCONRUN = 0;   /* Single conversion */
ADCTRL1 -> ADC2EN = 1;      /* Enable ADC2 */
ADCTRL1 -> ADC1EN = 1;      /* Enable ADC1 */

/* Resolver initializing */

*SYSCR &= 0xFF3F;          /* Make IOPC1 a IO Pin */
*PCDATDIR = 0x0303;        /* Make IOPC1 (FREEZE) = 1 */
                           /* Make IOPC0 (ENABLE) = 1 */

/* Set the COMPR registers for 50% duty cycle */

*COMPR1 = 1500;
*COMPR2 = 1500;
*COMPR3 = 1500;
*SCOMPR1 = 1500;
*SCOMPR2 = 1500;
*SCOMPR3 = 1500;

/* Initialize the DACs */

*PBDATDIR |= 0x7878;       /* Making IOPB3..6 high and outputs */

/* Initialize starting values */

PlqOutZ = 0;
PlqInZ = 0;
PidOutZ = 0;
PidInZ = 0;
IqBackZ = 0;
IdBackZ = 0;
Wait = 0;

IsRef = 0;
IsRefZ = 0;
limit = 260;
PISpeedInZ = 0;
PISpeedOutZ = 0;

*OCRB &= 0xFF0F;          /* Making IO1, IO2, IO3, IO4 active */
*PCDATDIR |= 0xF000;

```

```

/*=====*/
/*                                     Start of Main Program loop                                     */
/*=====*/

do
{
} while (*T1CNT > 5);

for(;;)
{
    *PCDATDIR |= 0x0080;

/*-----*/
    /* Read data in from ADC5 (Current Ia) and ADC10 (Current Ib) */

    ADCTRL1 -> ADC1CHSEL = 5;          /* Select ADC5 to be read */
    ADCTRL1 -> ADC2CHSEL = 2;          /* Select ADC10 to be read */
    ADCTRL1 -> ADCSOC = 1;              /* Begin conversion */
    while (ADCTRL1 -> ADCINTFLAG == 0); /* Wait for completion of conversion */
    ADCTRL1 -> ADCINTFLAG = 1;          /* Reset flag */
    IaBack = *ADC_FIFO1;                /* Read in data */
    IbBack = *ADC_FIFO2;

    IaBack = IaBack >> 6;                /* Scaling data to 16 bit value */
    IaBack &= 0x03FF;
    IaBack = IaBack - 512;

    IbBack = IbBack >> 6;
    IbBack &= 0x03FF;
    IbBack = IbBack - 514;

    /* Read data in from ADC6 (Current Ic) and ADC15 (Rotor speed) */

    ADCTRL1 -> ADC1CHSEL = 6;          /* Select ADC5 to be read */
    ADCTRL1 -> ADC2CHSEL = 7;          /* Select ADC15 to be read */
    ADCTRL1 -> ADCSOC = 1;              /* Begin conversion */
    while (ADCTRL1 -> ADCINTFLAG == 0); /* Wait for completion of conversion */
    ADCTRL1 -> ADCINTFLAG = 1;          /* Reset flag */
    IcBack = *ADC_FIFO1;                /* Read in data */
    Speed = *ADC_FIFO2;

    IcBack = IcBack >> 6;                /* Scaling data to 16 bit value */
    IcBack &= 0x03FF;
    IcBack = IcBack - 513;

    Speed = Speed >> 6;
    Speed &= 0x03FF;
    Speed = -(Speed - 512) + 2;          /* From 5V to 2.5V max. */

    /* Read data in from ADC1 (Is reference) and ADC11 (DC Bus-voltage) */

    ADCTRL1 -> ADC1CHSEL = 1;          /* Select ADC5 to be read */
    ADCTRL1 -> ADC2CHSEL = 3;          /* Select ADC10 to be read */
    ADCTRL1 -> ADCSOC = 1;              /* Begin conversion */
    while (ADCTRL1 -> ADCINTFLAG == 0); /* Wait for completion of conversion */
    ADCTRL1 -> ADCINTFLAG = 1;          /* Reset flag */
    SpeedRef = *ADC_FIFO1;              /* Read in data */
    GSBus = *ADC_FIFO2;

```

```

SpeedRef = SpeedRef >> 6;           /* Scaling of data */
SpeedRef &= 0x03FF;
SpeedRef = SpeedRef/4;              /* From 5V to 2.5V max. */

/* Reading the position of the rotor */

/* The mechanical rotor angle is 0 - 2047 to give a electrical rotor */
/* angle of 0 - 1023. The sine table will have a max. magnitude of 255 */

PCDATDIR &= 0xFFFD;                /* Make IOPC1 (FREEZE) = 0 */
PCDATDIR &= 0xFFFE;                /* Make IOPC0 (ENABLE) = 0 */
MrotorAngle = *Datain;
PCDATDIR |= 0x0003;                /* Make IOPC1 (FREEZE) = 1 */
                                   /* Make IOPC0 (ENABLE) = 1 */
MrotorAngle = MrotorAngle >> 5;    /* Make position a 16-bit value */
MrotorAngle &= 0x07FF;
MrotorAngle = MrotorAngle + 757;   /* Zero angle by adding a constant */
if (MrotorAngle > 2047)
    {MrotorAngle = MrotorAngle - 2047;}

ErotorAngle = MrotorAngle;         /* Calculate the electrical rotor angle */
if (ErotorAngle > 1023)
    {ErotorAngle = ErotorAngle - 1023;}

/*-----*/
/* Calculating the speed of the rotor */

if ((MrotorAngleZ > 1500) & (MrotorAngle < 500))
    {MrotorAngle = MrotorAngle + 2047;}
if ((MrotorAngleZ < 500) & (MrotorAngle > 1500))
    {MrotorAngleZ = MrotorAngleZ + 2047;}

SpeedBack = 10 * (MrotorAngle - MrotorAngleZ); /* Convert position to speed */

if (MrotorAngle > 2047)
    {MrotorAngle = MrotorAngle - 2047;}
if (MrotorAngleZ > 2047)
    {MrotorAngleZ = MrotorAngleZ - 2047;}
MrotorAngleZ = MrotorAngle;

ESpeedBack = (32 * SpeedBack) / 19; /* Scaling of Speed for decoupling */

/*-----*/
/* Calculation of angles for the Park-transformation */

angleA1 = ErotorAngle + 256;
if (angleA1 > 1023)
    {angleA1 = angleA1 - 1023;}
angleB1 = ErotorAngle - 341;
if (angleB1 < 0)
    {angleB1 = angleB1 + 1023;}
angleB2 = ErotorAngle - 85;
if (angleB2 < 0)
    {angleB2 = angleB2 + 1023;}
angleC1 = ErotorAngle + 341;
if (angleC1 > 1023)
    {angleC1 = angleC1 - 1023;}
angleC2 = ErotorAngle + 597;
if (angleC2 > 1023)

```



```

        {angleC2 = angleC2 - 1023;}

/*-----*/
/* Speed Controller */

    PISpeed = SpeedRef - SpeedBack;    /* Calculating the error */
    if (Wait < 6000)
        {Wait = Wait + 1;
         PISpeed = 0;
         IdRef = 102;}

    PISpeedIn = PISpeed;
    if ((IsRefZ >= limit) || (IsRefZ <= -limit)) /* Stop integrator in limit */
        {PISpeedIn = 0;}

    PISpeedOut = PISpeedOutZ + 6*(PISpeedIn - PISpeedInZ) + 0.009*PISpeedIn;
    /* U = U(Z-1) + Kp*[I - I(Z-1)] + Ki*T*I */
    PISpeedInZ = PISpeedIn;
    PISpeedOutZ = PISpeedOut;
    IsRef = (int) PISpeedOut;

    if ((IsRefZ >= limit) || (IsRef <= -limit))
        {IsRef = IsRef + 6*PISpeed ;}
    IsRefZ = IsRef;

    if (IsRef >= limit)
        {IsRef = limit;}
    if (IsRef <= -limit)
        {IsRef = -limit;}

    DAC1 = SpeedRef;
    DAC2 = SpeedBack;
    DAC3 = PISpeed;
    DAC4 = IqRef;

/*-----*/
/* Transforming Is and the angle to Id and Iq */

    if (Wait < 3000)
        {IdRef = 0;
         IsRef = 0;}
    IqRef = IsRef;

/*-----*/
/* abc-to-qdo transformation (Park-1) */

    IaBack = IaBack / 4;
    IbBack = IbBack / 4;
    IcBack = IcBack / 4;
    IqBack = (IaBack*sine(angleA1) )/256 + (IbBack*sine(angleB2))/256 +
    (IcBack*sine(angleC2))/256;
    IdBack = (IaBack*sine(ErotorAngle))/256 + (IbBack*sine(angleB1))/256 +
    (IcBack*sine(angleC1))/256;
    IqBack = 4 * (IqBack*2/3);
    IdBack = 4 * (IdBack*2/3);

/*-----*/

```

```

/* PI Controllers */

PidIn = IdRef - IdBack;          /* D-as Current error */
PidOut = PidOutZ + 9.99*(PidIn - PidInZ) + 0.0099*PidIn;
/* U = U(Z-1) + Kpd*(I - I(z-1)) + Kid*T*I */

PidOutZ = PidOut;               /* Saving data for next loop */
PidInZ = PidIn;
Vd = (int) PidOut;

PlqIn = IqRef - IqBack;          /* Q-as Current error */
PlqOut = PlqOutZ + 4.95*(PlqIn - PlqInZ) + 0.05*PlqIn;
/* U = U(Z-1) + Kpq*(I - I(z-1)) + Kiq*T*I */

PlqOutZ = PlqOut;               /* Saving data for next loop */
PlqInZ = PlqIn;
Vq = (int) PlqOut;

/*-----*/
/* Decoupling of Md */

dlq_dt = IqBack - IqBackZ;
IqBackZ = IqBack;
if (IdBack < 0)
    {MdInd = -1*(md(-IdBack));}
else
    {MdInd = md(IdBack);}
MdDec = (MdInd * dlq_dt * -1) / 16;

/* Decoupling of Mq */

dld_dt = IdBack - IdBackZ;
IdBackZ = IdBack;
if (IqBack < 0)
    {MqInd = -1*(mq(-IqBack));}
else
    {MqInd = mq(IqBack);}
MqDec = (MqInd * dld_dt * -1) / 16;

/*-----*/
/* Decoupling of Speed Voltages */

if (IdBack < 0)
    {SpeedVolD = ((-1 * fluxd(-1*IdBack)) * ESspeedBack) / 32;}
else
    {SpeedVolD = (fluxd(IdBack) * ESspeedBack) / 32;}

if (IqBack < 0)
    {SpeedVolQ = ((-1 * fluxq(-1*IqBack)) * ESspeedBack) / 32;}
else
    {SpeedVolQ = (fluxq(IqBack) * ESspeedBack) / 32;}

/*-----*/
/* Decoupling of terms */

Vd = Vd + MdDec - SpeedVolQ;    /* Decouple the Md inductance */
Vq = Vq + MqDec + SpeedVolD;    /* Decouple the Mq inductance */

/*-----*/

```

```

/* qdo-to-abc transformation (Park) */

Vd = Vd / 16;
Vq = Vq / 16;
Va = (Vd*sine(ErotorAngle))/256 + (Vq*sine(angleA1))/256;
Vb = (Vd*sine(angleB1) )/256 + (Vq*sine(angleB2))/256;
Vc = (Vd*sine(angleC1) )/256 + (Vq*sine(angleC2))/256;
Va = 16*Va;
Vb = 16*Vb;
Vc = 16*Vc;

/*-----*/
/* Giving the values to the PWM part of the DSP */

*COMPR1 = 1500 + Va;
*COMPR2 = 1500 + Vb;
*COMPR3 = 1500 + Vc;
*SCOMPR1 = 1500 - Va;
*SCOMPR2 = 1500 - Vb;
*SCOMPR3 = 1500 - Vc;

/*-----*/
/* Write to the DACs */

*PBDATDIR &= 0xFFF7; /* Select DAC 1 */
*DataOut = DAC1*8 + 1000;
*PBDATDIR |= 0x0008;

*PBDATDIR &= 0xFFEF; /* Select DAC 2 */
*DataOut = DAC2*8 + 1000;
*PBDATDIR |= 0x0010;

*PBDATDIR &= 0xFFDF; /* Select DAC 3 */
*DataOut = DAC3*8 + 1000;
*PBDATDIR |= 0x0020;

*PBDATDIR &= 0xFFBF; /* Select DAC 4 */
*DataOut = DAC4*4 + 1000;
*PBDATDIR |= 0x0040;

*PCDATDIR &= 0xFF7F;
do
{
    } while (*T1CNT > 5);
}

```


Appendix C – DSP Program for Constant-current- angle Speed Control

```

/*****
/*      SpeedControlConstAngle.c (14 January 2002)      */
/*      */
/*      (Control program for PI speed control of the RSM) */
/* The switching is unipolar with a switching frequency of 3.33kHz. */
*****/

#include "F240.h"

volatile unsigned int *EVIMRA    = (volatile unsigned int *) 0x742C;
volatile unsigned int *ACTR      = (volatile unsigned int *) 0x7413;
volatile unsigned int *SACTR     = (volatile unsigned int *) 0x7414;
volatile unsigned int *DBTCON    = (volatile unsigned int *) 0x7415;
volatile unsigned int *COMPR1    = (volatile unsigned int *) 0x7417;
volatile unsigned int *COMPR2    = (volatile unsigned int *) 0x7418;
volatile unsigned int *COMPR3    = (volatile unsigned int *) 0x7419;
volatile unsigned int *SCOMPR1   = (volatile unsigned int *) 0x741A;
volatile unsigned int *SCOMPR2   = (volatile unsigned int *) 0x741B;
volatile unsigned int *SCOMPR3   = (volatile unsigned int *) 0x741C;
volatile unsigned int *COMCON    = (volatile unsigned int *) 0x7411;
volatile unsigned int *GPTCON    = (volatile unsigned int *) 0x7400;
volatile unsigned int *T1CON     = (volatile unsigned int *) 0x7404;
volatile unsigned int *DataIn    = (volatile unsigned int *) 0x8000;
volatile unsigned int *DataOut   = (volatile unsigned int *) 0x8001;

int  IaBack, IbBack, IcBack, IdBack, IqBack;
int  IsRef, IdRef, IqRef, Wait;
int  Va, Vb, Vc, Vd, Vq, limit, IsRefZ;
int  DAC1, DAC2, DAC3, DAC4, MdInd, MqInd, MrotorAngleZ, SpeedBack;
int  MrotorAngle, ErotorAngle, CurrentAngle, CurrentAngle2;
int  angleA1, angleB1, angleB2, angleC1, angleC2, Speed, GSBUS, SpeedBack;
int  QOut, DOut, Time, TimeTemp, PITemp, Inductance, SpeedTemp;
int  dlq_dt, dld_dt, IqBackZ, IdBackZ, MdDec, MqDec;
int  PISpeed, PISpeedIn, PISpeedInZ, SpeedRef, ESPEEDBACK, SpeedVolD, SpeedVolQ;
float PISpeedOut, PISpeedOutZ;
float PIqOutZ, PIqOut, PIqInZ, PIqIn, PIdOutZ, PIdOut, PIdInZ, PIdIn;
extern int  sine(), md(), mq(), fluxd(), fluxq();

void main(void)
{
    /* Initialize the interrupt routines */

    *EVIMRA |= 0x0001; /* Enable the PDPINT interrupt routine */

    /* Initialize the PWM and dead time routines */

    *ACTR  = 0x0999; /* Make PWM1, PWM3 and PWM5 active low */
                /* Make PWM2, PWM4 and PWM6 active high */
    *SACTR = 0x0015; /* Make PWM7, PWM8 and PWM9 active low */
    *DBTCON = 0x0000; /* Dead time = 0us, all 3 outputs active */
    *COMCON = 0x0307; /* Initialize Compare PWM mode */
    *COMCON = 0x8307; /* Initialize Compare PWM mode */

    /* Initialize the counters */

    *T1PR = 3000; /* Period = 2*(50ns x 3000) = 300us */
    *T1CNT = 0; /* Starting value for counter = 0 */
    *T1CON = 0x0A802; /* Initialize Counter #1 */
    *T1CON = 0x0A842; /* Initialize Counter #1 */

```

```

/* Initialize the ADs */

*OCRA |= 0x070F;          /* Make ADC0, ADC1, ADC8 and ADC9 active */
ADCTRL2 -> ADCPSCALE = 0; /* Pre-scale value = 0 */
ADCTRL2 -> ADCEVSOC = 0;
ADCTRL2 -> ADCEXTSOC = 0;
ADCTRL1 -> suspend_free = 0;
ADCTRL1 -> suspend_soft = 1;
ADCTRL1 -> ADCIMSTART = 0;
ADCTRL1 -> ADCINTEN = 0; /* Disable interrupt */
ADCTRL1 -> ADCCONRUN = 0; /* Single conversion */
ADCTRL1 -> ADC2EN = 1; /* Enable ADC2 */
ADCTRL1 -> ADC1EN = 1; /* Enable ADC1 */

/* Resolver initializing */

*SYSCR &= 0xFF3F;          /* Make IOPC1 a IO Pin */
*PCDATDIR = 0x0303;        /* Make IOPC1 (FREEZE) = 1 */
                          /* Make IOPC0 (ENABLE) = 1 */

/* Set the COMPR registers for 50% duty cycle */

*COMPR1 = 1500;
*COMPR2 = 1500;
*COMPR3 = 1500;
*SCOMPR1 = 1500;
*SCOMPR2 = 1500;
*SCOMPR3 = 1500;

/* Initialize the DACs */

*PBDATDIR |= 0x7878;      /* Making IOPB3..6 high and outputs */

/* Initialize starting values */

PlqOutZ = 0;
PlqInZ = 0;
PidOutZ = 0;
PidInZ = 0;
IqBackZ = 0;
IdBackZ = 0;
Wait = 0;

IsRef = 0;
IsRefZ = 0;
limit = 260;
PISpeedInZ = 0;
PISpeedOutZ = 0;

*OCRB &= 0xFF0F;          /* Making IO1, IO2, IO3, IO4 active */
*PCDATDIR |= 0xF000;

```



```

/*=====*/
/*                                     Start of Main Program loop                                     */
/*=====*/

do
{
} while (*T1CNT > 5);

for(;;)
{
    *PCDATDIR |= 0x0080;

/*-----*/
    /* Read data in from ADC5 (Current Ia) and ADC10 (Current Ib) */

    ADCTRL1 -> ADC1CHSEL = 5;          /* Select ADC5 to be read */
    ADCTRL1 -> ADC2CHSEL = 2;          /* Select ADC10 to be read */
    ADCTRL1 -> ADCSOC = 1;             /* Begin conversion */
    while (ADCTRL1 -> ADCINTFLAG == 0); /* Wait for completion of conversion */
    ADCTRL1 -> ADCINTFLAG = 1;         /* Reset flag */
    IaBack = *ADC_FIFO1;               /* Read in data */
    IbBack = *ADC_FIFO2;

    IaBack = IaBack >> 6;              /* Scaling data to 16 bit value */
    IaBack &= 0x03FF;
    IaBack = IaBack - 512;

    IbBack = IbBack >> 6;
    IbBack &= 0x03FF;
    IbBack = IbBack - 514;

    /* Read data in from ADC6 (Current Ic) and ADC15 (Rotor speed) */

    ADCTRL1 -> ADC1CHSEL = 6;          /* Select ADC5 to be read */
    ADCTRL1 -> ADC2CHSEL = 7;          /* Select ADC15 to be read */
    ADCTRL1 -> ADCSOC = 1;             /* Begin conversion */
    while (ADCTRL1 -> ADCINTFLAG == 0); /* Wait for completion of conversion */
    ADCTRL1 -> ADCINTFLAG = 1;         /* Reset flag */
    IcBack = *ADC_FIFO1;               /* Read in data */
    Speed = *ADC_FIFO2;

    IcBack = IcBack >> 6;              /* Scaling data to 16 bit value */
    IcBack &= 0x03FF;
    IcBack = IcBack - 513;

    Speed = Speed >> 6;                /* Scaling data */
    Speed &= 0x03FF;
    Speed = -(Speed - 512) + 2;        /* From 5V to 2.5V max. */

    /* Read data in from ADC1 (Is reference) and ADC11 (DC Bus-voltage) */

    ADCTRL1 -> ADC1CHSEL = 1;          /* Select ADC5 to be read */
    ADCTRL1 -> ADC2CHSEL = 3;          /* Select ADC10 to be read */
    ADCTRL1 -> ADCSOC = 1;             /* Begin conversion */
    while (ADCTRL1 -> ADCINTFLAG == 0); /* Wait for completion of conversion */
    ADCTRL1 -> ADCINTFLAG = 1;         /* Reset flag */
    SpeedRef = *ADC_FIFO1;             /* Read in data */
    GSBUS = *ADC_FIFO2;

    SpeedRef = SpeedRef >> 6;          /* Scaling of data */

```

```

SpeedRef &= 0x03FF;
SpeedRef = SpeedRef/4;                                /* From 5V to 2.5V max. */

/* Reading the position of the rotor */

/* The mechanical rotor angle is 0 - 2047 to give a electrical rotor */
/* angle of 0 - 1023. The sine table will have a max. magnitude of 255 */

PCDATDIR &= 0xFFFD;                                    /* Make IOPC1 (FREEZE) = 0 */
PCDATDIR &= 0xFFFE;                                    /* Make IOPC0 (ENABLE) = 0 */
MrotorAngle = *Datain;
PCDATDIR |= 0x0003;                                    /* Make IOPC1 (FREEZE) = 1 */
                                                    /* Make IOPC0 (ENABLE) = 1 */
MrotorAngle = MrotorAngle >> 5;                        /* Make position a 16-bit value */
MrotorAngle &= 0x07FF;
MrotorAngle = MrotorAngle + 757;                        /* Zero angle by adding a constant */
if (MrotorAngle > 2047)
    {MrotorAngle = MrotorAngle - 2047;}

ErotorAngle = MrotorAngle;                             /* Calculate the electrical rotor angle */
if (ErotorAngle > 1023)
    {ErotorAngle = ErotorAngle - 1023;}

/*-----*/
/* Calculating the speed of the rotor */

if ((MrotorAngleZ > 1500) & (MrotorAngle < 500))
    {MrotorAngle = MrotorAngle + 2047;}
if ((MrotorAngleZ < 500) & (MrotorAngle > 1500))
    {MrotorAngleZ = MrotorAngleZ + 2047;}

SpeedBack = 10 * (MrotorAngle - MrotorAngleZ);        /* Convert position to speed */

if (MrotorAngle > 2047)
    {MrotorAngle = MrotorAngle - 2047;}
if (MrotorAngleZ > 2047)
    {MrotorAngleZ = MrotorAngle - 2047;}
MrotorAngleZ = MrotorAngle;

ESpeedBack = (32 * SpeedBack) / 19;                    /* Scaling of Speed for decoupling */

/*-----*/
/* Calculation of angles for the Park-transformation */

angleA1 = ErotorAngle + 256;
if (angleA1 > 1023)
    {angleA1 = angleA1 - 1023;}
angleB1 = ErotorAngle - 341;
if (angleB1 < 0)
    {angleB1 = angleB1 + 1023;}
angleB2 = ErotorAngle - 85;
if (angleB2 < 0)
    {angleB2 = angleB2 + 1023;}
angleC1 = ErotorAngle + 341;
if (angleC1 > 1023)
    {angleC1 = angleC1 - 1023;}
angleC2 = ErotorAngle + 597;
if (angleC2 > 1023)
    {angleC2 = angleC2 - 1023;}

```



```

/*-----*/
/* Speed Controller */

PISpeed = SpeedRef - SpeedBack; /* Calculating the error */
if (Wait < 6000)
    {Wait = Wait + 1;
    PISpeed = 0;}

PISpeedIn = PISpeed;
if ((IsRefZ >= limit) || (IsRefZ <= -limit)) /* Stop integrator in limit */
    {PISpeedIn = 0;}

PISpeedOut = PISpeedOutZ + 6*(PISpeedIn - PISpeedInZ) + 0.009*PISpeedIn;
/* U = U(Z-1) + Kp*[I - I(Z-1)] + Ki*T*I */
PISpeedInZ = PISpeedIn;
PISpeedOutZ = PISpeedOut;
IsRef = (int) PISpeedOut;

if ((IsRefZ >= limit) || (IsRef <= -limit))
    {IsRef = IsRef + 6*PISpeed ;}
IsRefZ = IsRef;

if (IsRef >= limit)
    {IsRef = limit;}
if (IsRef <= -limit)
    {IsRef = -limit;}

DAC1 = SpeedRef;
DAC2 = SpeedBack;
DAC3 = PISpeed;
DAC4 = IsRef;

/*-----*/
/* Transforming Is and the angle to Id and Iq */

if (Wait < 3000)
    {IsRef = 0;}
CurrentAngle = 193;
CurrentAngle2 = 449;
if (IsRef < 0)
    {CurrentAngle = 830;
    CurrentAngle2 = 63;
    IsRef = IsRef * -1;}

IsRef = IsRef / 4;
IqRef = 4 * ((IsRef * sine(CurrentAngle)) / 256);
IdRef = 4 * ((IsRef * sine(CurrentAngle2)) / 256);

/*-----*/
/* abc-to-qdo transformation (Park-1) */

IaBack = IaBack / 4;
IbBack = IbBack / 4;
IcBack = IcBack / 4;
IqBack = (IaBack*sine(angleA1) )/256 + (IbBack*sine(angleB2))/256 +
(IcBack*sine(angleC2))/256;
IdBack = (IaBack*sine(ErotorAngle))/256 + (IbBack*sine(angleB1))/256 +
(IcBack*sine(angleC1))/256;
IqBack = 4 * (IqBack*2/3);
IdBack = 4 * (IdBack*2/3);

```



```

/*-----*/
/* PI Controllers */

PIdIn = IdRef - IdBack;          /* D-as Current error */
PIdOut = PIdOutZ + 9.99*(PIdIn - PIdInZ) + 0.0099*PIdIn;
/* U = U(Z-1) + Kpd*(I - I(z-1)) + Kid*T*I */

PIdOutZ = PIdOut;               /* Saving data for next loop */
PIdInZ = PIdIn;
Vd = (int) PIdOut;

PIqIn = IqRef - IqBack;         /* Q-as Current error */
PIqOut = PIqOutZ + 4.95*(PIqIn - PIqInZ) + 0.05*PIqIn;
/* U = U(Z-1) + Kpq*(I - I(z-1)) + Kiq*T*I */

PIqOutZ = PIqOut;               /* Saving data for next loop */
PIqInZ = PIqIn;
Vq = (int) PIqOut;

/*-----*/
/* Decoupling of Md */

dlq_dt = IqBack - IqBackZ;
IqBackZ = IqBack;
if (IdBack < 0)
    {MdInd = -1*(md(-IdBack));}
else
    {MdInd = md(IdBack);}
MdDec = (MdInd * dlq_dt * -1) / 16;

/* Decoupling of Mq */

dId_dt = IdBack - IdBackZ;
IdBackZ = IdBack;
if (IqBack < 0)
    {MqInd = -1*(mq(-IqBack));}
else
    {MqInd = mq(IqBack);}
MqDec = (MqInd * dId_dt * -1) / 16;

/*-----*/
/* Decoupling of Speed Voltages */

if (IdBack < 0)
    {SpeedVolD = ((-1 * fluxd(-1*IdBack)) * ESspeedBack) / 32;}
else
    {SpeedVolD = (fluxd(IdBack) * ESspeedBack) / 32;}

if (IqBack < 0)
    {SpeedVolQ = ((-1 * fluxq(-1*IqBack)) * ESspeedBack) / 32;}
else
    {SpeedVolQ = (fluxq(IqBack) * ESspeedBack) / 32;}

/*-----*/
/* Decoupling of terms */

Vd = Vd + MdDec; /*- SpeedVolQ;*/ /* Decouple the Md inductance */
Vq = Vq + MqDec; /*+ SpeedVolD;*/ /* Decouple the Mq inductance */

```

```

/*-----*/
    /* qdo-to-abc transformation (Park) */

    Vd = Vd / 16;
    Vq = Vq / 16;
    Va = (Vd*sine(ErotorAngle))/256 + (Vq*sine(angleA1))/256;
    Vb = (Vd*sine(angleB1) )/256 + (Vq*sine(angleB2))/256;
    Vc = (Vd*sine(angleC1) )/256 + (Vq*sine(angleC2))/256;
    Va = 16*Va;
    Vb = 16*Vb;
    Vc = 16*Vc;

/*-----*/
    /* Giving the values to the PWM part of the DSP */

    *COMPR1 = 1500 + Va;
    *COMPR2 = 1500 + Vb;
    *COMPR3 = 1500 + Vc;
    *SCOMPR1 = 1500 - Va;
    *SCOMPR2 = 1500 - Vb;
    *SCOMPR3 = 1500 - Vc;

/*-----*/
    /* Write to the DACs */

    *PBDATDIR &= 0xFFFF7;      /* Select DAC 1 */
    *DataOut = DAC1*8 + 1000;
    *PBDATDIR |= 0x0008;

    *PBDATDIR &= 0xFFEF;      /* Select DAC 2 */
    *DataOut = DAC2*8 + 1000;
    *PBDATDIR |= 0x0010;

    *PBDATDIR &= 0xFFDF;      /* Select DAC 3 */
    *DataOut = DAC3*8 + 1000;
    *PBDATDIR |= 0x0020;

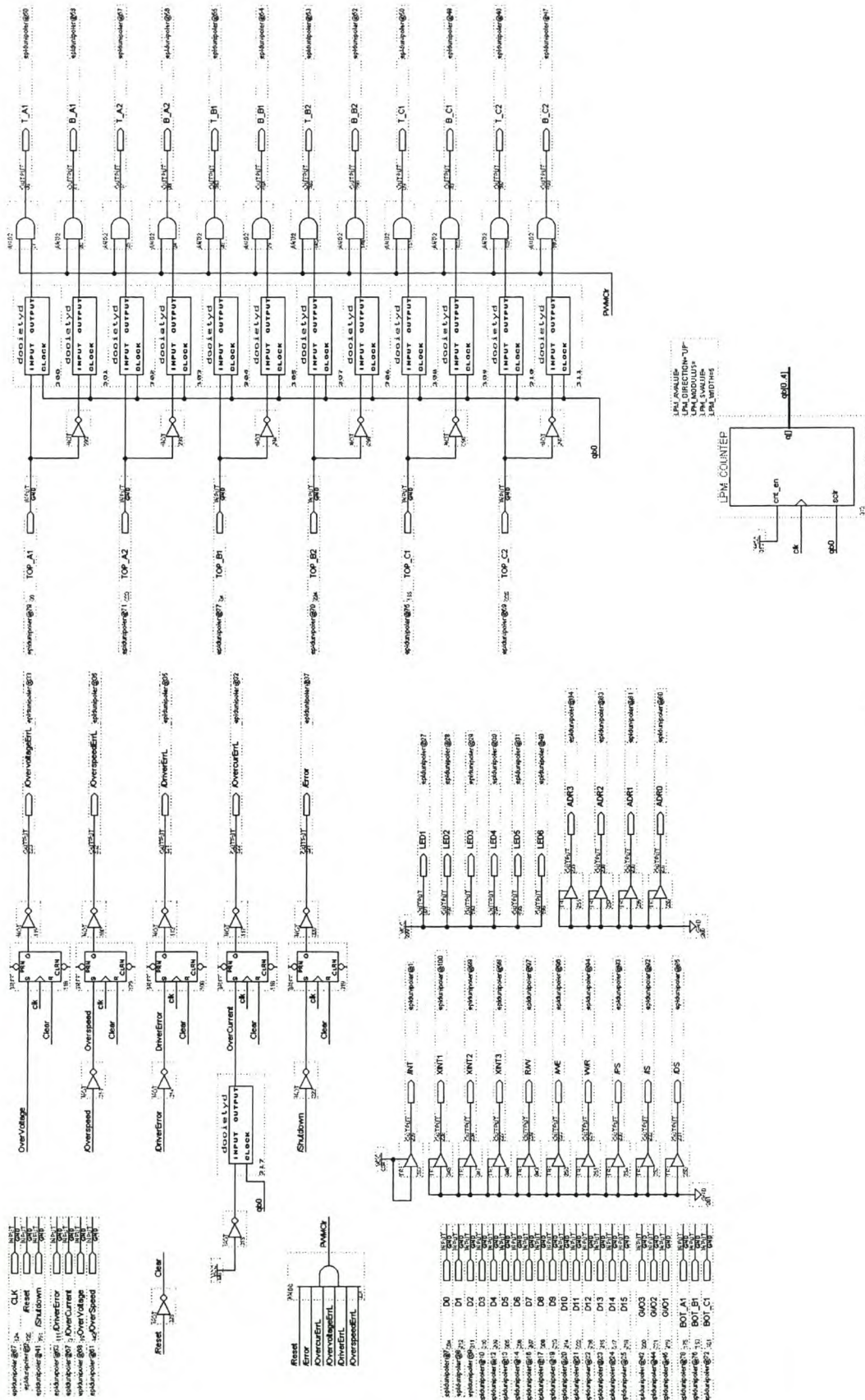
    *PBDATDIR &= 0xFFBF;      /* Select DAC 4 */
    *DataOut = DAC4*4 + 1000;
    *PBDATDIR |= 0x0040;

    *PCDATDIR &= 0xFF7F;
    do
    {
        } while (*T1CNT > 5);
    }
}

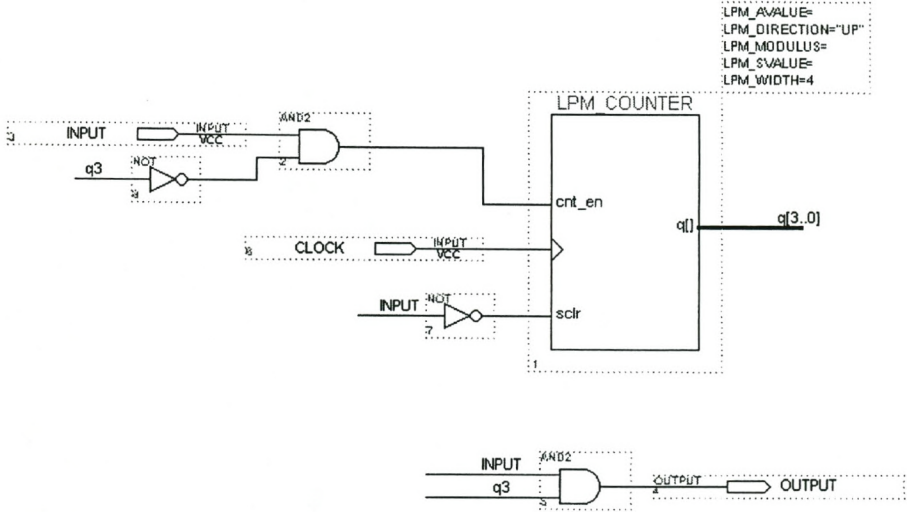
```

Appendix D – EPLD Program

Protection / Unipolar Program



Dead time Program



Appendix E – Finite Element Program of RSM


```

c *****
c      PROGRAM: Calculate flux linkages with skew
c *****

c Phase currents as a function of current angle "alpha" and rotor
c position "theta-m" in mechanical degrees from phase a magnetic axis
c anti-clockwise
c -----

c Set the off-set rotor-angle with the rotor q-axis on the magnetic
c axis of phase a
      th_off=-15.0d0*pi/180.0d0

c For skew over one slot pitch, ks=5; for unskew ks=1.
c For average over slot pitch, ka=5, otherwise ka=1
      ks=1
      ka=5

c Slice displacement:
      d_slice=1.0d0*(2.0d0)*pi/dble(n_st_st)/dble(ks)

c STEP 1: Get phase current, dq-currents and current angle:
do id = 0,200,40
  do iq = 0,200,10
    cur_id = dble(id)
    cur_iq = dble(iq)
    alpha = atan(cur_iq/(cur_id+1.0d-12))
    c_peak = sqrt(cur_id**2 + cur_iq**2)/dble(np_cir)

    print *, '===== Verwysingswaardes ====='
    print *, 'c_peak = ', c_peak, ' [A]'
    print *, 'Current angle = ', alpha*180.0d0/pi, ' [Degrees]'
    print *, 'cur_iq = ', cur_iq, ' [A]'
    print *, 'cur_id = ', cur_id, ' [A]'

c STEP 2: Rotor position and/or rotate rotor
do irotate=0,0,40
  rot_ang=0.5d0*dble(irotate)*pi/180.0d0

c Set average torque and average flux to zero before starting
      Ta = 0.0d0
      Ttot = 0.0d0
      lamda_dm=0.0
      lamda_qm=0.0

      do i=1,3
        flinka(i)=0.0d0
      end do

      do ik=1,3
        do jk=1,3
          mss(ik,jk)=0.0
        end do
      end do

c STEP 3: Average over slot pitch? Yes: ka=5; No: ka=1
c (Note that in this case the current phasor moves with the rotor)

do ithet_m=1,ka
  if (ka.eq.5) then
    theta_m=dble(ithet_m-ka+2)*d_slice+rot_ang
  else if (ka.eq.1) then
    theta_m=rot_ang
  end if

  c_cur(1)=c_peak * dsin(nppole*theta_m+alpha)
  c_cur(2)=c_peak * dsin(nppole*theta_m+alpha-2.0d0*pi/3.0d0)
  c_cur(3)=c_peak * dsin(nppole*theta_m+alpha+2.0d0*pi/3.0d0)

  cur(1)=c_cur(1)
  cur(2)=c_cur(2)
  cur(3)=c_cur(3)

```

```

c  print *, '
c  print *, 'cur(1) = ', cur(1), ' [A]'
c  print *, 'cur(2) = ', cur(2), ' [A]'
c  print *, 'cur(3) = ', cur(3), ' [A]'
c  print *, 'angle = ', angle*180.0d0/pi, ' [Degrees]'

      angle = theta_m * nppole
      call park(cur(1),cur(2),cur(3),angle,curq,curd)
      curq = curq * dble(np_cir)
      curd = curd * dble(np_cir)

      print *, '
      print *, '===== Berekende waardes ====='
      print *, 'curq,cur_iq = ', curq,cur_iq, ' [A]'
      print *, 'curd,cur_id = ', curd,cur_id, ' [A]'

c Initialize before skew:
      T = 0.0d0
      T100 = 0.0d0

      do i=1,3
        flink(i)=0.0
      end do

      do ik=1,3
        do jk=1,3
          mss(ik,jk)=0.0
        end do
      end do

c STEP 4: Skew the machine? Yes: ks=5; No: ks=1.
c      (Note that in this case the current phasor is stationary)
      do i=1,ks
        if (ks.eq.1) then
          dtheta=theta_m+th_off
          alpha_i=alpha
        else if (ks.eq.5) then
          dtheta=dbl(i-ks+2)*d_slice+theta_m+th_off
          alpha_i=dbl(3-i)*d_slice*nppole+alpha
        end if

        call step(dtheta,sraz,nfour,terms,an,bn,nrs,nt,
+ ah,ap,bh,bp,la,thao)

        call nonlinear(neq,nelm,nnode,nprof,nraz,v0,ar,nd,node,
+ razpnt,itype,jdiag,s,sraz,rel,c_cur,a,x,y,razind,
+ nppole,
+ acoil,nc_turns,w,np,ns,zp,
+ nlines,line,neigh,bcur)

c just do position 1 again for accuracy
      if (i.eq.1) then
        call nonlinear(neq,nelm,nnode,nprof,nraz,v0,ar,nd,node,
+ razpnt,itype,jdiag,s,sraz,rel,c_cur,a,x,y,razind,
+ nppole,
+ acoil,nc_turns,w,np,ns,zp,
+ nlines,line,neigh,bcur)
      end if

c a is magnetic vector potential
c      do ii=1,nnode
c        atorq(ii)=a(ii)
c      end do

c calculate torque
      call torque(rrad,srad,torq1,100,nppole,nrs,nraz,ax,
& a,an,bn,razind)
      T100 = T100 + torq1
      call torque(rrad,srad,torq1,1,nppole,nrs,nraz,ax,
& a,an,bn,razind)
      T = T + torq1

c -----
c Air-gap flux densities, flux linkages and induce EMF are now
c determined of the fundamental. On Razek airgap points the

```

```

c Bgap(radial) = 1/r.dA/dtheta is calculated and then the
c fundamental Bgap using Foerier analysis. Ephase contains
c the value obtained by using the fundamental Bgap
c -----
      open (10, FILE="results/bgap.res", form='formatted')
      rewind 10
      do ib=(nrs+1),nt
        bgap(ib)=1/(st_id/2.)/(dabs(rthet(ib+1)-rthet(ib))) *
+ (A(razind(ib+1))-A(razind(ib)))
        write (10,125) ((rthet(ib+1)+rthet(ib))/2.0),bgap(ib)
      end do
125 format(2F10.5)
      close (10,STATUS='KEEP')

c get Fourier coefficients for fundamental
      nhar=1
      bxs=0.0d0
      bys=0.0d0
      nppole=np/2
      do if=(nt+1),(nrs+2),-1
        bi=bgap(if-1)

c general method used with same result:
        del_theta=nppole*(rthet(if-1)-rthet(if))
        theta_p=nppole*((rthet(if)+rthet(if-1))/2.0d0-pi/2.0d0)
        bxs=bx+bi*dsin(theta_p)*del_theta
        bys=bys+bi*dcos(theta_p)*del_theta
      end do

      am=bx*2.0d0/pi
      bm=bys*2.0d0/pi
      bpeak=dsqrt(am*am+bm*bm)

c get flux linkage space phasor angle:
      ang_i=atan(bm/(am+1.0d-12))-pi/2.0d0

c adjust according to electrical d-axis position of rotor
      ang_i=-ang_i-pi-nppole*dtheta

c test if valid
      if (alpha_i.gt.pi/4.0.and.ang_i.lt.0.0) ang_i=ang_i+pi
      if (alpha_i.lt.pi/4.0.and.ang_i.lt.-pi/4.0) ang_i=ang_i+pi

      phi=2.0*bpeak*st_id*w/dbl(np)/dbl(ks)
      lamda_d=phi*nphase*qwn*dcos(ang_i)
      lamda_q=phi*nphase*qwn*dsin(ang_i)
      lamda_dm=lamda_dm+lamda_d
      lamda_qm=lamda_qm+lamda_q

c -----
c Flux Linkage Psi(i) is calculated using the integral of A.dl
c -----

      do ic2=1,3
        Psi(ic2)=0.0
        Do iii=1,nelm
          if (abs(itype(iii)).eq.ic2) then
            Psi(ic2)=Psi(ic2) +
+ (a(node(iii,1))+a(node(iii,2))+a(node(iii,3)))*ar(iii)
+ *sign(1,itype(iii))
          end if
        end do
        psi(ic2)=2.0d0*nppole*w*nc_turns(ic2)*Psi(ic2)/3.0d0/acoil(ic2)
        flink(ic2)=flink(ic2)+psi(ic2)
      end do

      end do
c END OF SKEW (ks)

      Ta=(T/ks)+Ta
      Ttot=(T100/ks)+Ttot
      do i=1,3
        flinka(i)=(flink(i)/ks)+flinka(i)
      end do

```



```

        end do
c END OF AVERAGE (ka)

c =====
c                               FINAL CALCULATIONS
c =====

c calculate: from airgap flux:

    Ta = Ta/dbl(ka)
    Ttot = Ttot/dbl(ka)

    lamda_s=dsqrt(lamda_dm**2+lamda_qm**2)
    fluxang=datan(lamda_qm/(lamda_dm+1.0d-12))
    if (alpha.gt.pi/4.0.and.fluxang.lt.0.0) fluxang=fluxang+pi
    fl_am=lamda_s*dsin(nppole*theta_m+fluxang)

    fluxang=fluxang*180.0d0/pi
    Ephase=2.0*pi*50./sqrt2*lamda_s

c calculate the dq-axis inductances from airgap flux:

    l_d=lamda_dm/(cur_id+1.0d-12)
    l_q=lamda_qm/(cur_iq+1.0d-12)
    Te=3.0d0*(l_d-l_q)*cur_id*cur_iq

    If (cur_iq.eq.0.0) l_q=0.0
    If (cur_id.eq.0.0) l_d=0.0

c -----
c calculate d- and q-axis L, flink, Erms (p. ...)
c -----

    do ikk=1,3
        flink(ikk)=flinka(ikk)/dbl(ka)/dbl(np_cir)
    end do

c Co-phasal 3rd harmonic flux linkages can be obtained
c from the actual three phase flux linkages
c phase a : 1 / phase b : 2 / phase c : 3
    fl_a3 = (flink(1) + flink(2) + flink(3))/3.0d0
    fl_b3 = (flink(1) + flink(2) + flink(3))/3.0d0
    fl_c3 = (flink(1) + flink(2) + flink(3))/3.0d0

c Calculate the fundamental flux linkages
    fl_a1 = flink(1) - fl_a3
    fl_b1 = flink(2) - fl_b3
    fl_c1 = flink(3) - fl_c3

c Calculate the dq axis flux
    angle = rot_ang * nppole
    call park(fl_a1,fl_b1,fl_c1,angle,fl_q,fl_d)
        call park(flink(1),flink(2),flink(3),angle,fl_q,fl_d)

    print *, '
    print *, '===== Berekenende Vloeddigtheid ====='
    print *, 'id,iq,T = ',cur_id,cur_iq,Ta,' [A], [Nm]'
    print *, 'fld,flq = ',fl_d,fl_q,' [Weber Turns]'

    fl_Tsize = dsqrt(fl_q**2 + fl_d**2)
    fl_Tangle = datan(fl_q/fl_d)
    fl_Tangle = fl_Tangle*180.0d0/pi
c    print *, 'fl_Tsize = ',fl_Tsize,' [Weber Turns]'
c    print *, 'fl_Tangle = ',fl_Tangle,' [Degrees]'

c -----

    l_leak=(flink(1)-fl_am)/(c_cur(1)*np_cir)+0.001d0
    l_d=l_d+l_leak
    l_q=l_q+l_leak

c    fl_ab=flink(1)-flink(3)
c    fl_bc=flink(3)-flink(2)
c    fl_ca=flink(2)-flink(1)

```

```

c      xf=nppole*theta_m+30.0d0*pi/180.0d0
c      yf=nppole*theta_m-90.0d0*pi/180.0d0
c      beta=atan((fl_ab*dsin(yf)-fl_bc*dsin(xf))/(fl_bc*dcos(xf)
c + -fl_ab*dcos(yf)))
c      f_peak=fl_ab/(dsqrt(3.0d0)*dsin(xf+beta))
c      l_df=1000.0d0*(f_peak*dcos(beta)/cur_id+l_end)
c      l_qf=1000.0d0*(f_peak*dsin(beta)/cur_id+l_end)
c      Tef=3.0d0*(l_df-l_qf)*cur_id*cur_id/1000.0d0
c      e_d=f_peak*dcos(beta+pi/2.0d0)*omegas
c      e_q=f_peak*dsin(beta+pi/2.0d0)*omegas

      Tef=Te
      e_d=-l_q*cur_id*omegas
      e_q=l_d*cur_id*omegas
      e_rms=dsqrt(e_d*e_d+e_q*e_q)/sqrt2

c -----
c calculate iron core loss resistance r_c (pp. 55-58)
c -----
c
c calculate mass of teeth and yoke:
      ar_slot=2.0d0*acoil(1)*50.0/48.0+0.5*(st_gw+st_tw)*st_shs+
+ st_gw*st_sht
      mass_y=7880.0d0*w*(pi/4.0)*(st_od**2-(st_id+2.0*st_slh)**2)
      mass_t=7880.0d0*w*((pi/4.0)*((st_id+2.0*st_slh)**2-st_id**2)-
+ n_st_st*ar_slot)

c calculate maximum flux density in teeth and yoke:
      flux_pp=e_rms/(4.44d0*f*nphase*qwn)
      bp_air=flux_pp*dblnp)/(2.0d0*st_id*w)
      t_pitch=pi*st_id/a_slt
      bp_t=bp_air*t_pitch/t_wid
      bp_y=flux_pp/(2.0d0*st_yh*w)

c calculate iron core loss and core loss resistance:
      p_core=0.0337d0*(f**1.32)*((bp_t**2)*mass_t+(bp_y**2)*mass_y)
      r_c=3.0d0*e_rms**2/p_core

c -----
c      calculate kVA, powerfactor, efficiency
c -----

      cur_idl=cur_id+e_d/r_c
      cur_idl=cur_id+e_q/r_c
      cur_ang=atan(cur_idl/(cur_id+1.0d-12))*180.0d0/pi
      i_rms=dsqrt(cur_idl**2+cur_id**2)/sqrt2
      v_d=e_d-omegas*cur_idl*(l_end)+r_s*cur_idl
      v_q=e_q+omegas*cur_idl*(l_end)+r_s*cur_idl
      v_rms=dsqrt(v_d*v_d+v_q*v_q)/sqrt2
      kva=3.0d0*i_rms*v_rms/1000.0d0
      p_fact=dcos(atan(dabs(v_d/v_q))+atan(cur_idl/cur_idl))
      p_out=omegas*Tef/dblnp)-p_wf
      T_out=Tef-p_wf*dblnp)/omegas
      p_cu=3.0d0*i_rms**2*r_s
      p_in=kva*p_fact*1000.0d0
      eff=p_out/p_in
      cur_den=i_rms/(a_sc*1.0d+6*dblnp_cir))
      p_loss=p_cu+p_core+p_wf
      p_loss=p_cu+p_core

c -----
c results to main program
c -----

      Tr=640.0d0
      pry=10500.0d0

c      print *,'
c      print *,' fl_a3 = fl_b3 = flc3 = ',fl_a3
c      print *,' fl_a1 = ',fl_a1
c      print *,' fl_b1 = ',fl_b1
c      print *,' fl_c1 = ',fl_c1
c      print *,' ===== '

c      ypar(1) = c_peak*dblnp_cir)
c      ypar(2) = alpha*180.0d0/pi

```

```

c      ypar(3) = fl_Tsize
c      ypar(4) = fl_Tangle
c      write(17,132) ypar(1), ypar(2), ypar(3), ypar(4)

c write to file:performb.res

      ypar(1) = rot_ang*180.0d0/pi
      ypar(2) = cur_id
      ypar(3) = fl_d
      ypar(4) = cur_iq
      ypar(5) = fl_q

      write(22,131) ypar(1),ypar(2),ypar(3),ypar(4),ypar(5)

c      write(22,103)
c      write(22,131) ypar(1),ypar(2),ypar(3),ypar(4),ypar(5),ypar(6),
c      +      ypar(7),ypar(8),ypar(9),ypar(10),ypar(11)

      end do
c END OF ROTATE ROTOR (irotate, rot_ang)

      write(22,101)
      end do
c END OF IQ

      write(22,101)
      end do
c END OF ID

      close (22,status='keep')
      close (17,status='keep')
c
c store data and results for fpm:
c
      out_file='bug.fpl'
      call save_dth(out_file,nde,nelm,nnode,x,y,node,ittype,
      + nlines,line,neigh,a,dtheta,rr)

      goto 133

c      stop

c *****
c-----
c This is handy to get inductances as a function of rotor
c position. Set ind_do=1 to make it calculate
c-----
c
c calculate inductances if requested
c

      if (ind_do.eq.1) then
c
c open all the files:
c
      open (8, file="ind/mainind.ind", form='formatted')
      rewind 8
      open (9, file="ind/m11.ind", form='formatted')
      open (10, file="ind/m22.ind", form='formatted')
      open (11, file="ind/m33.ind", form='formatted')
      open (12, file="ind/m12.ind", form='formatted')
      open (13, file="ind/m13.ind", form='formatted')
      open (14, file="ind/m23.ind", form='formatted')
      open (15, file="ind/m21.ind", form='formatted')
      open (16, file="ind/m31.ind", form='formatted')
      open (17, file="ind/m32.ind", form='formatted')
c
c do until time = peroid+4*tmdt
c
      time=0.0d0
      tmdt=period/72.0d0
c have to start with induc at t-4*tmdt

      time=0.0d0-4.0d0*tmdt
      dtheta=0.0

```



```

        tmst=time
        tmet=period+5.0*tmdt

        icount=1
        do while (icount.le.143)

c
c get the mesh in the correct position:
c
        call step(dtheta,sraz,nfour,terms,an,bn,nrs,nt,
+ ah,ap,bh,bp,la,thao)
c
c do a linear solve
c and inductances extracted
c

        call lim_lin(neq,nelm,nnode,nprof,nraz,v0,ar,nd,node,
+ razpnt,itype,jdiag,s,sraz,rel,a,x,y,zp,nppole,
+ acoil,nc_turns,w,ms,np,ns,
+ nlines,line,neigh)

        write (9,*) dtheta,ms(1,1)
        write (10,*) dtheta,ms(2,2)
        write (11,*) dtheta,ms(3,3)
        write (12,*) dtheta,ms(1,2)
        write (13,*) dtheta,ms(1,3)
        write (14,*) dtheta,ms(2,3)
        write (15,*) dtheta,ms(2,1)
        write (16,*) dtheta,ms(3,1)
        write (17,*) dtheta,ms(3,2)
c
c store the calculated inductances
c
        write (8,*) dtheta
c        print*,icount = ',icount
        do i=1,ns
        do j=1,ns
        mainind(icount,i,j)=ms(i,j)
        write (8,*) mainind(icount,i,j)
        end do
        end do

        icount=icount+1
        time=time+tmdt
        dtheta=dtheta+pi/nppole/72.0d0

        end do
c
c end do for inductance calculations
c
        close (8,status='keep')

        end if
c
c finished calculating inductances.
c Read them in if requested, and calculated:
c

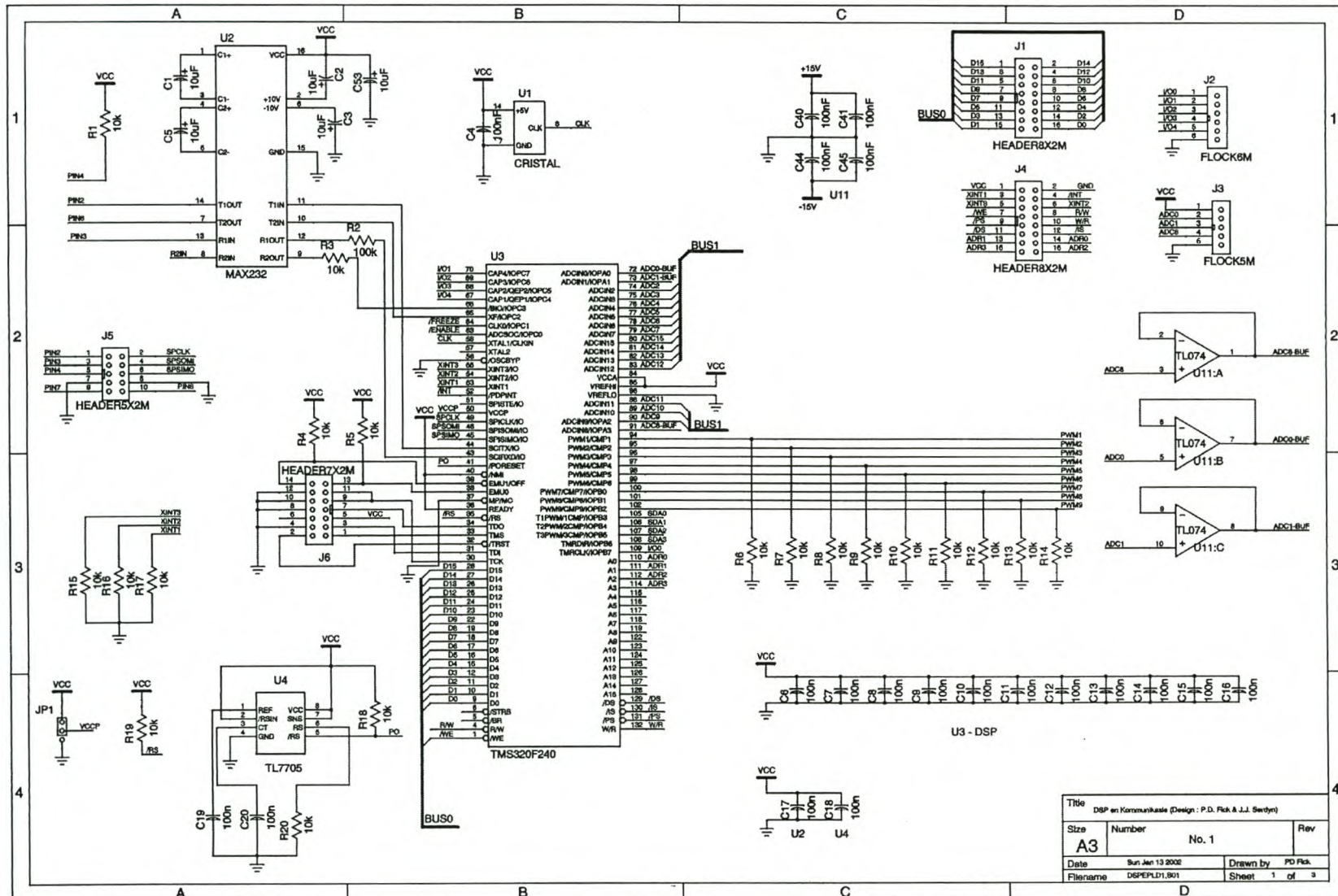
133 continue
        RETURN
        END

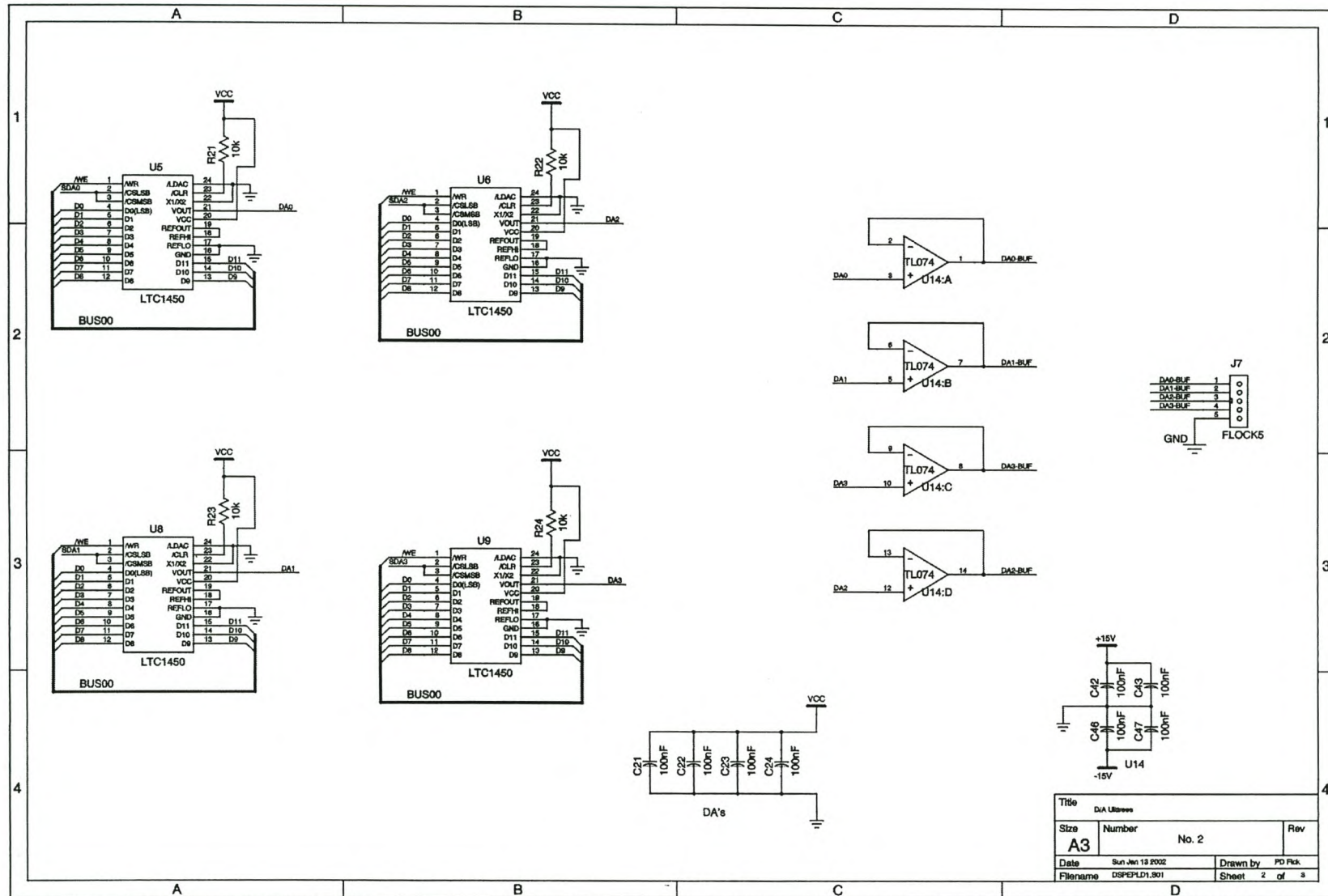
c
c end of eesolve program
c
c *****

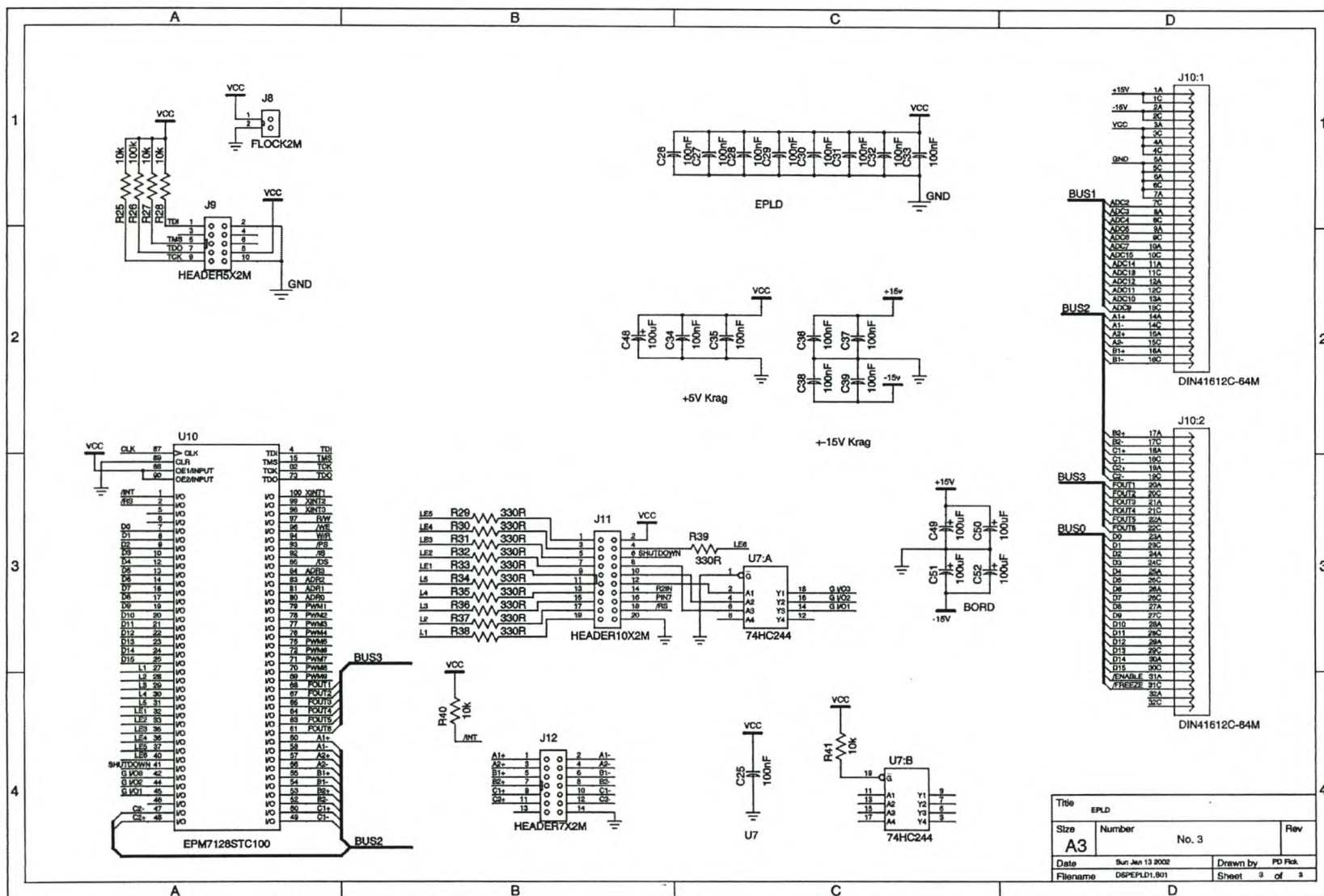
```

Appendix F – Schematic diagrams of the DSP controller

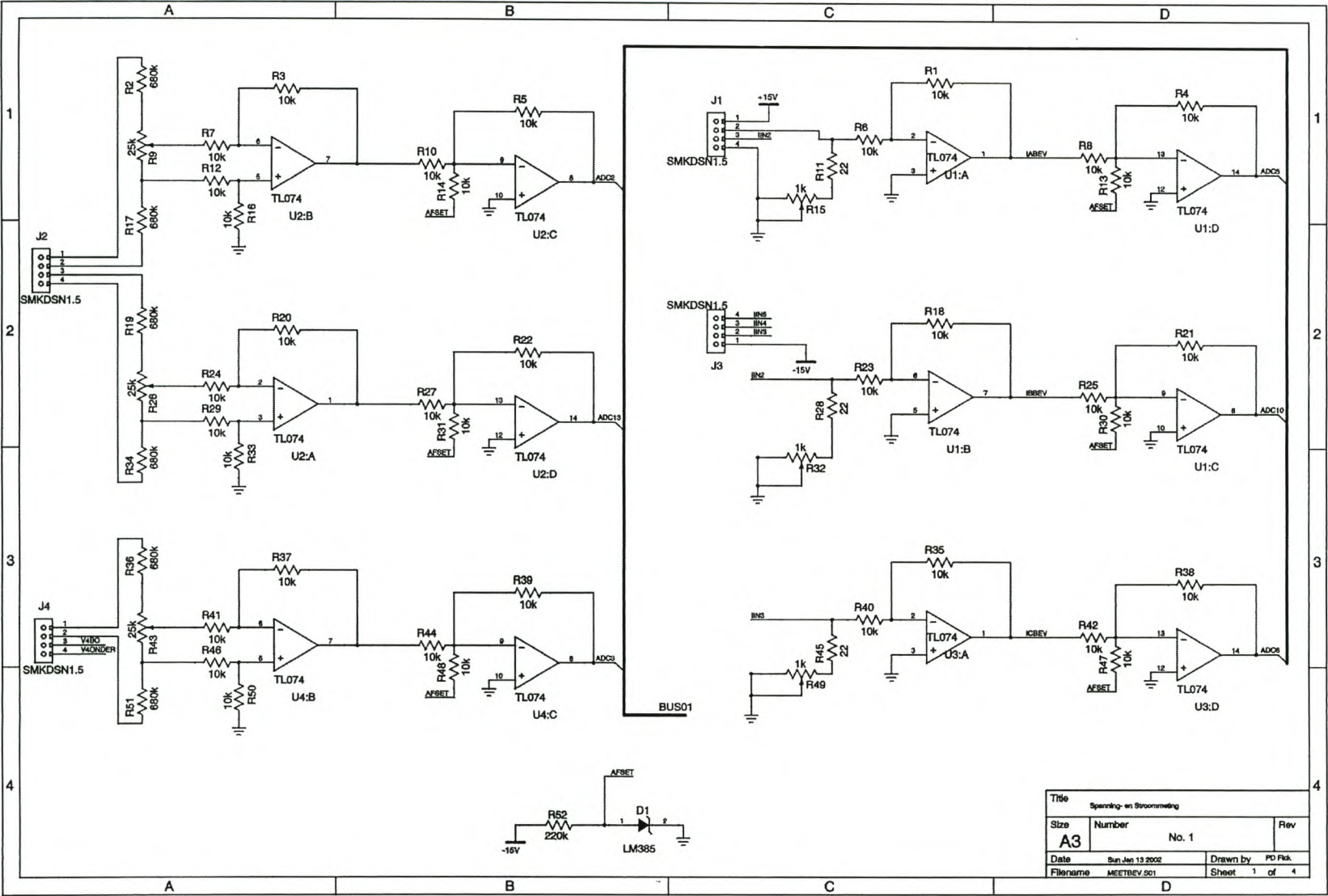
Processor/Protection Card

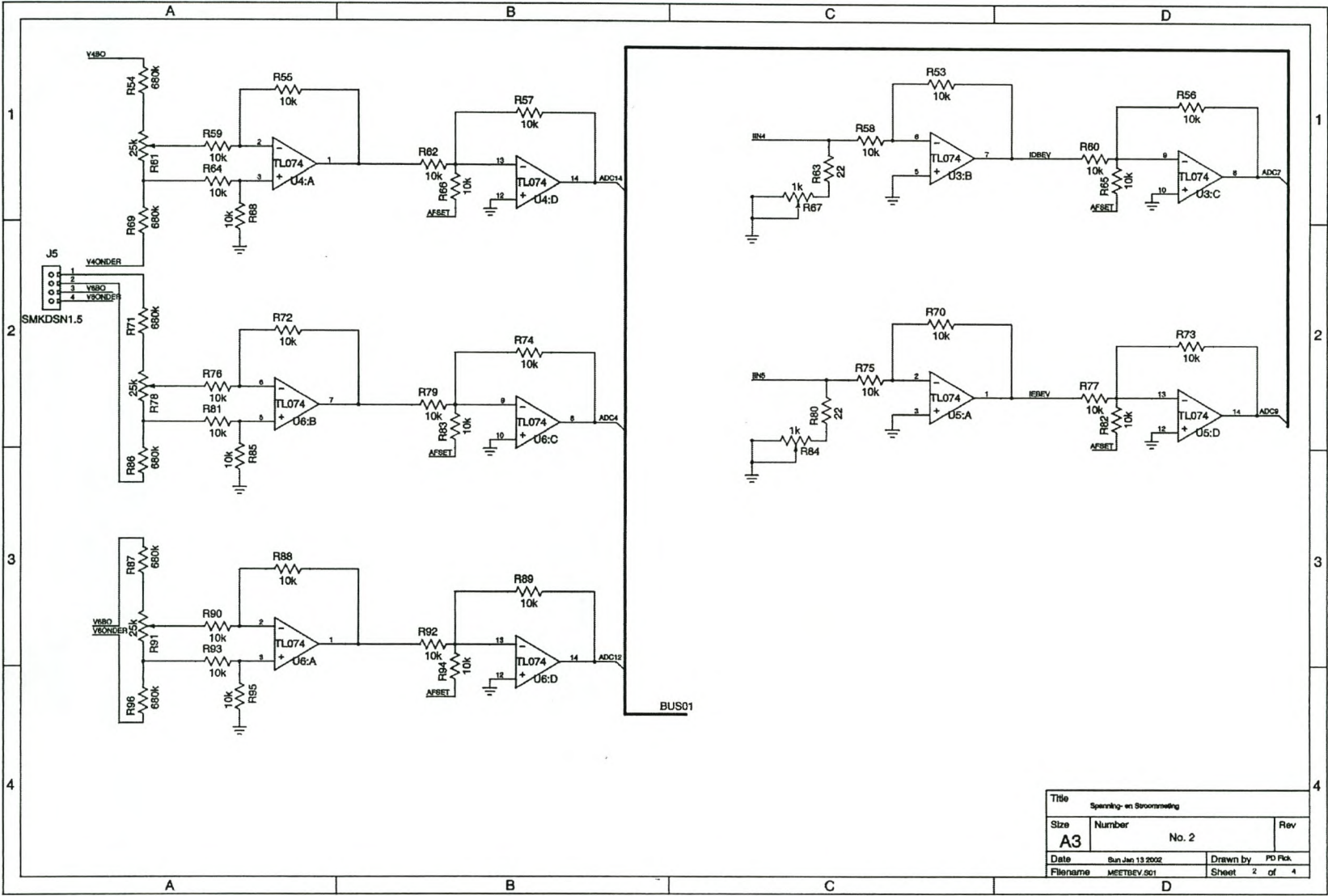


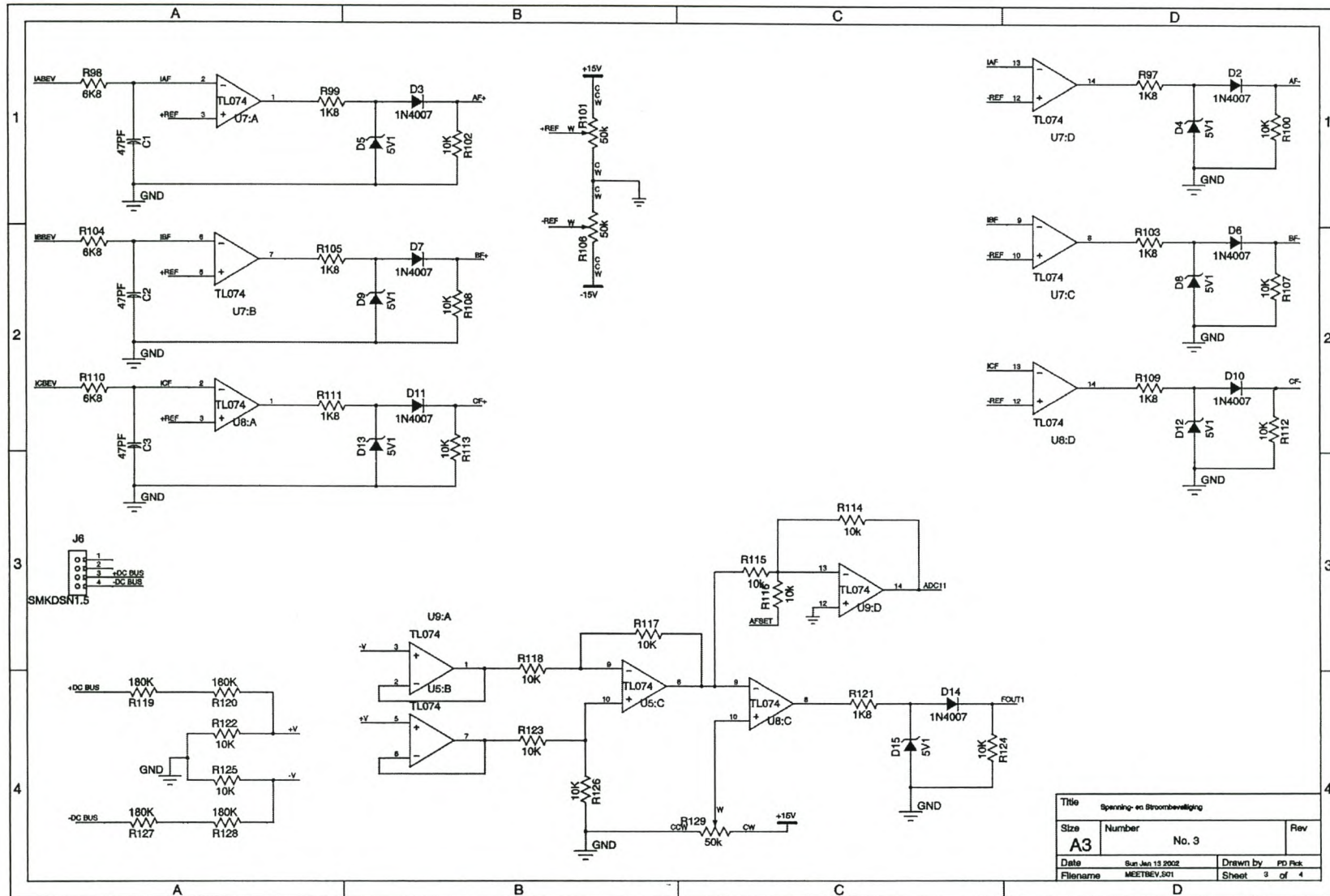


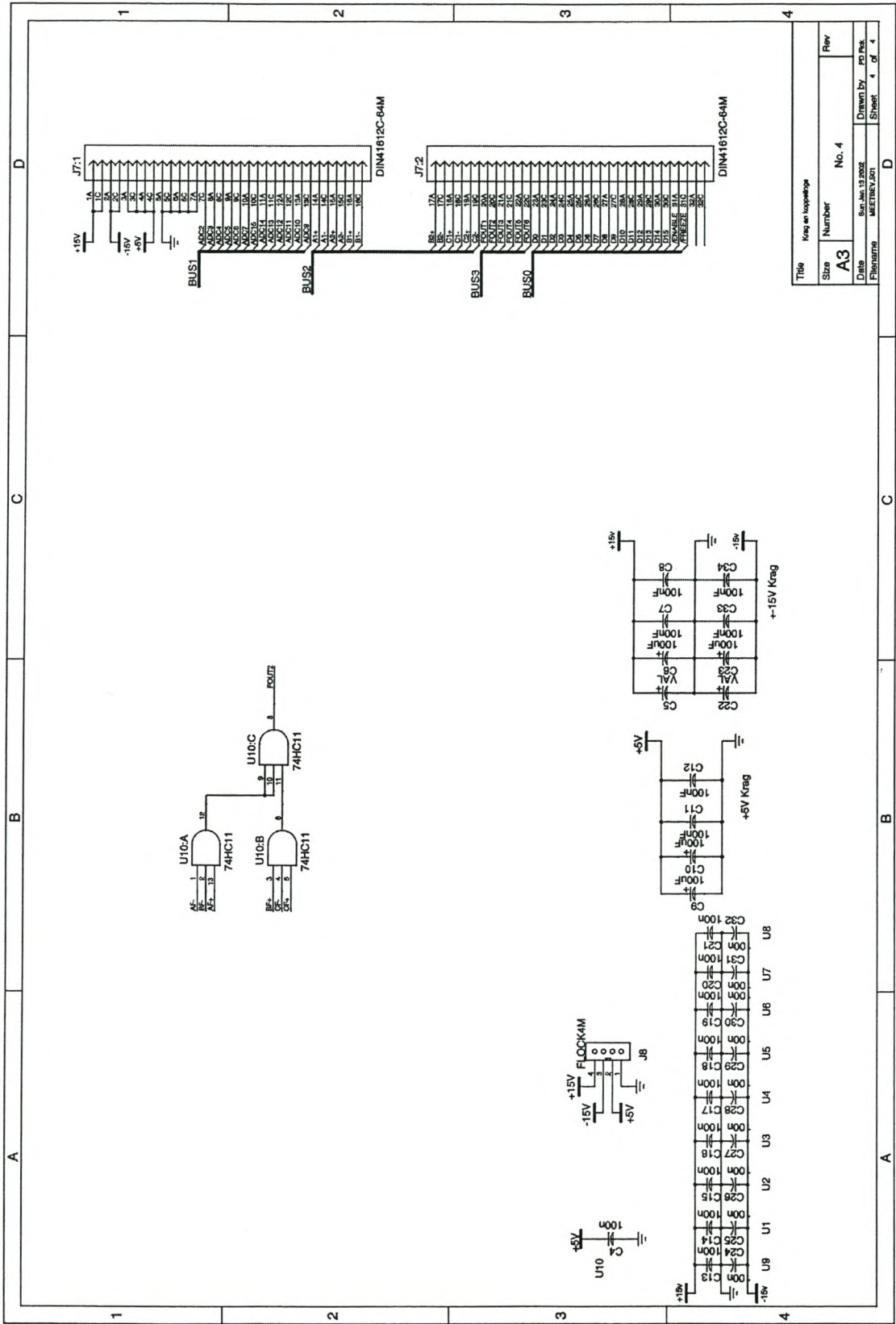


Measurement Card

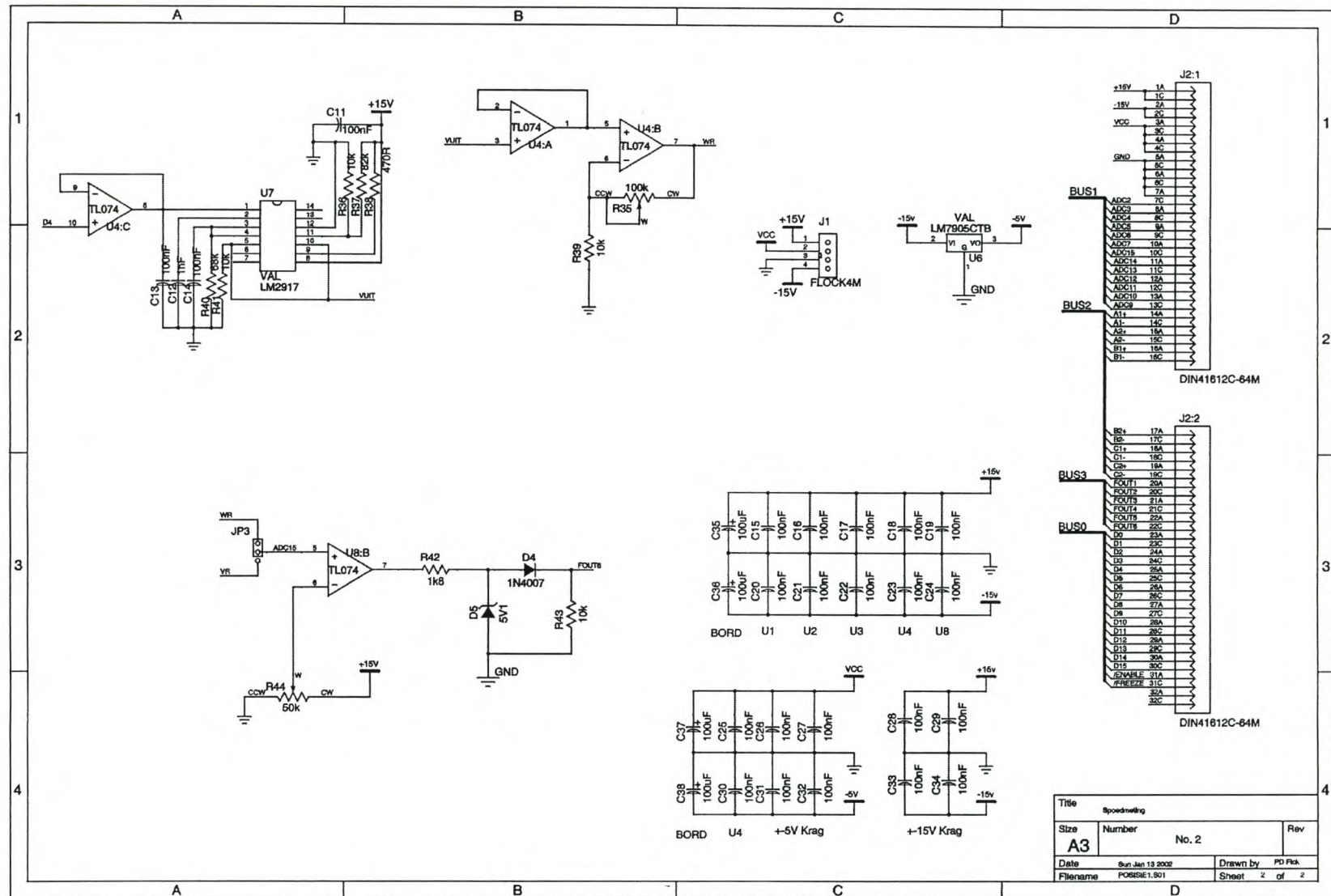




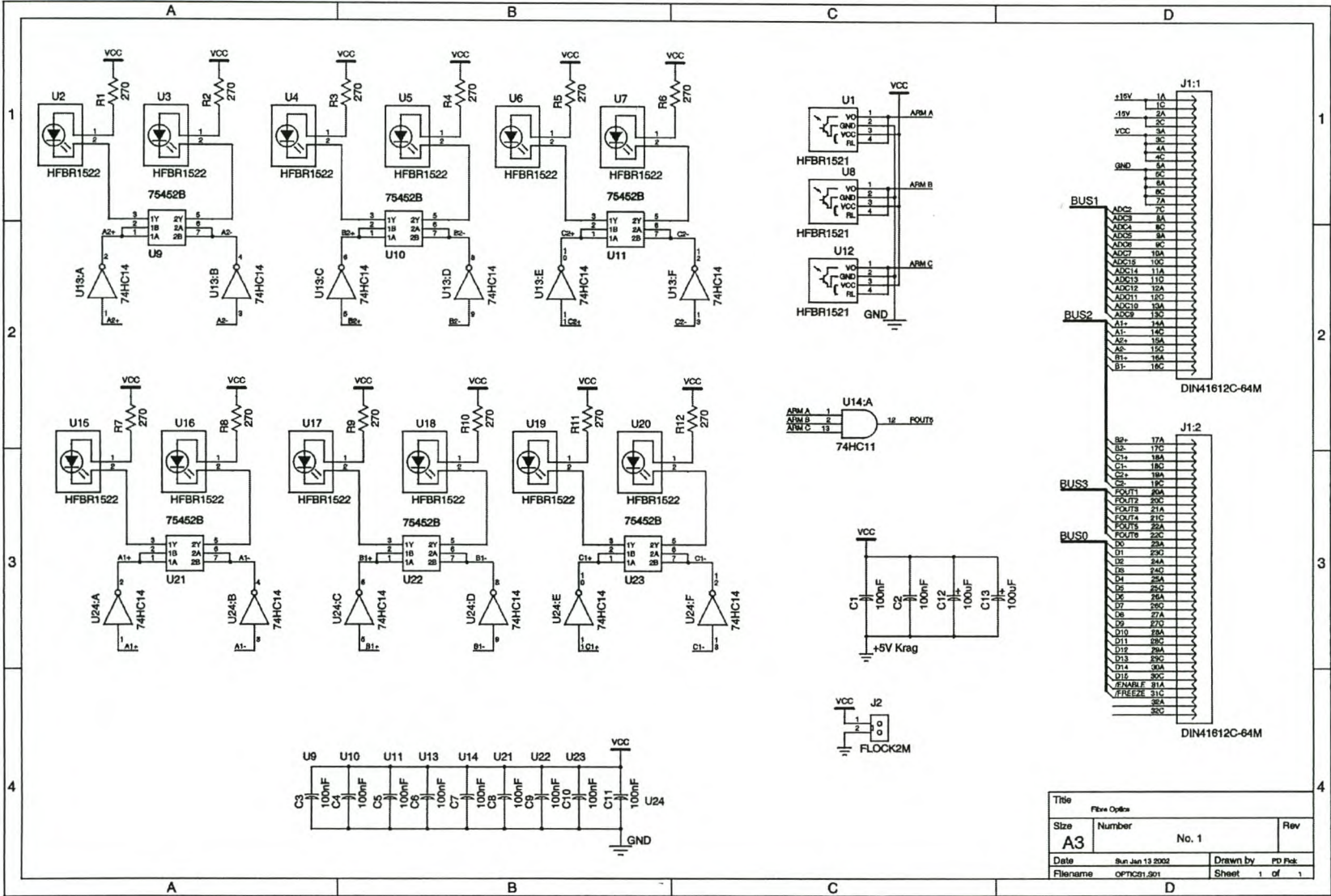








Fibre Optic Interface Card



Title				Back Plane van Eurokas			
Size		Number				Rev	
A3		No. 1					
Date				Sun Jan 13 2002		Drawn by	
Filename				BACKP.B01		P.D. Fick	
				Sheet		1 of 1	

